DE MONTFORT UNIVERSITY

SIMULATION AND IMPLEMENTATION OF ROTOR FLUX CONTROL FOR AN INDUCTION MOTOR

by

ANCA NOVINSCHI

A Thesis submitted in partial fulfilment of the requirements of
De Montfort University for the
Degree of Doctor of Philosophy

DEPARTMENT OF
ELECTRICAL AND ELECTRONIC ENGINEERING

September 1998

Declaration of Author's Rights

Acknowledgements

I would like to express my special thanks to Professor McCormick for his invaluable help advice and tireless encouragements. Also I am very grateful to Dr. W.F.Low for his help and for his always smiling face.

Thanks are also expressed to the Electrical and Electronic Department of De Montfort University for the award of the scholarship without which this PhD work could have never become reality. My thanks are due to the above mentioned Department for providing the necessary facilities.

Many thanks to Mrs. Sheila Hayto for her kind help and invaluable support through the whole period of the PhD.

Special thanks go to Tim O'Mara, Steve Gillett, Steve Regester, Ian Reynolds, Manbir Sambhi and Pritesh Karia who always encouraged me, helped me and tried to keep me cheerful. I am also very grateful to Cal English from *Mechanical Department* who built the test-rig as I designed it. I would also like to thank Andy Rylott, the Sun station `master`. I am grateful to Jeen G. Khor for our technical and non-technical talks and for his availability to do `some tests` whenever I wished and needed. I would like to mention my colleague Ahmed Abouarkoub for his help in practical electronics. Thanks are due to Andrei Dinu who helped me with useful `C programming` tips.

Very special thanks to all research and technical staff at University of L'Aquila, Faculty of Engineering, Italy (*Dipartimento di Ingegneria Elettrica, Universita degli studi di L'Aquila*). I would like to explicitly thank : Professore Enzo Chiricozzi, Professore Franco Parasiliti, Ing. Marco Tursini, Ing. Riccardo de Gabriele and the technicians: Romualdo Tiberio, Fabrizio Mancini, Gianni Cirella, Carlo Masciovecchio, Achile Spaziani. Very special gratitude is expressed towards my great friends: Ing. Antonio Ometto, Ing. Giovanni Conti and Ing. Roberto Petrella.

Finally, and not least, I am very grateful to my parents (Elena and Alexandru Novinschi) and to my friend (Florian Ivan) in Romania, for their patience and understanding. They always supported me even when being neglected, most of the time.

Abstract

The objective of this research is to model an induction motor drive and to investigate a vector control strategy for the induction motor.

The induction motor control techniques are reviewed and vector control of the induction motor is introduced. Subsequently, the induction motor is briefly described and the different electrical equivalent circuits models used for analysis are presented.

The DQ-axis theory is established as the most appropriate for modelling the transient response of induction motor drives. Therefore this theory is adopted and a model for the induction motor is achieved using the Simulink toolbox within MATLAB software. This model is general, however the results presented are for a specific inverter/induction motor drive system. Speed and torque responses of the motor to various input conditions, like step or different level inertia loads are presented. A combined model of the induction motor and the inverter is derived in order to simulate the behaviour of the system.

The induction motor model is extended to cater for general periodic input excitations and the PWM inverter model is combined with the induction motor model to simulate the drive system. This is realised by decomposing the PWM output into harmonics and combining together a number of harmonic fed motors.

It is shown that vector control technique using spatial frames produces expressions for the electromagnetic torque similar to those of a d.c. machine. This implies that induction motor control can be realised by the decoupled control of the flux-producing component and torque-producing component of the stator current, similar to controlling the field and

armature currents in a d.c. machine. The space phasor theory is presented and the principle of vector control, oriented with the rotor-flux vector, is explained.

Parameter estimation procedures, simulated using Pascal language, have been developed and values are estimated using voltage and current pulses applied to the motor. Employing an already assembled test rig, at the *Department of Electrical Engineering, University of L'Aquila, Italy*, practical tests were carried out and procedures were implemented for a control scheme based on SAB 80C166 microcontroller.

Further work reported includes the realisation of a test rig based on an induction motor of 1.5 kW. This was carried out at the *Department of Electrical and Electronic Engineering, De Montfort University*. Finally the vector control scheme was implemented for the TMS320C240 Evaluation Module and test results obtained which are presented and critically commented upon.

Table of contents

Chapter 1

Introduction

1.1 Introduction - Market Information

Electric motor technology is more than *100* years old, however some technologies developed in the late *1800s* are still used today in various configurations. Despite being very mature, a brilliant future is predicted for the motor industry thanks to advances in intelligent motor control. MTT (*Motion Tech Trends*) estimate the value of all motors manufactured in 1996 is of *US$ 47.5 billion* worldwide and by the year 2001 motor production is expected to reach *US$ 66 billion*. Production of three-phase induction motor represents *20.4%* of the total market while brushed d.c., universal, brushless and stepper motors represent more than *50%* [1].

Nowadays the electric motor is becoming a link between digital intelligence and the physical execution of the required work, consequently it is finding ever more use in all aspects of modern life. Automation of complex activities is now feasible due to the possibility of employing sophisticated sensors, intelligent software and microprocessors in conjunction with electric motors in industrial, commercial and consumer equipment [1].

The range and typical application areas for the most common type of electric drive, the induction motor, are briefly reviewed.

1.2 Induction Motor Control - General Review

D.C motors have been extensively used in variable speed applications mainly because they are easy to control. Control can be effected by both varying the armature voltage or by controlling field current. There are certain disadvantages associated with using d.c. motors, the main being the need for periodic maintenance due to the existence of the commutator and brushes. In addition, due to the likelihood of sparking at the rotating contacts they cannot be used in corrosive or explosive environments.

Induction motors are used instead of d.c. machines wherever possible because they are, in general smaller for the same rating, are robust and less expensive. However the control of a.c. motors is more complex than that for d.c. motors. The basic methods for controlling induction motors may be summarised as: resistance control, slip-energy recovery scheme, both applicable to wound-rotor machines, pole-changing, pole-amplitude modulation, supply-voltage variation and variable frequency-variable voltage [2].

The simplest type of speed control for wound-rotor induction motors may be realised using external resistances in series with the rotor circuit. For a required value of speed, the corresponding slip can be evaluated and consequently the necessary external resistance calculated. If a wide range of speed is required then a variable resistor of 0 to 100 times the rotor resistance may be necessary. However this method has the disadvantages associated with heat dissipation and energy loss. As a consequence of these drawbacks, slip-energy recovery schemes were developed. In these schemes the energy normally dissipated in the external resistance is fed back into the 3-phase supply of the machine.

For applications where dual-speeds are required, pole-changing and PAM techniques are available. Using this approach requires the induction machine to have either two separate three-phase stator windings with different numbers of pole-pairs or in the PAM technique. developed by G.H. Rawcliffe [3], a way to achieve multiple sets of poles in a single stator winding. In PAM the resulting number of poles can be in ratios other than 2:1 and the

method achieves a cost reduction when compared with the method of using two separate windings.

Speed control can also be realised by supply voltage variation. To obtain a variable voltage, a switched a.c. supply using inverse-parallel thyristors can be used. The main disadvantage is the creation of harmonic voltages and currents which increase losses and produce harmonic torques. The control method is restricted to speed variation of about 0.2 p.u. and machines rated up to several kilowatts.

A refined control method is obtained by using a variable-voltage variable-frequency supply. Maintaining a constant ratio of $\frac{V}{f}$ produces near constant flux conditions within the motor except at very low speeds. Implementation of this method is realised by employing an inverter drive or a cycloconverter. The cycloconverter transforms a three-phase supply at power frequency to one at lower frequency by the appropriate switching and natural commutation of the control devices. The several disadvantages associated with this approach include: complicated control circuitry, harmonic generation and high cost. However, improved switching strategies is making the use of cycloconverters more practical. Currently it is mostly used for large drives of *1 MW* and upwards.

Inverter drives can be categorised into current source inverters and voltage source inverters. The current source inverter drive is robust since the control strategy inherently limits overload currents and therefore offers short-circuit protection. By reducing the inverter frequency, the speed is bigger than synchronous speed and slip is negative. The machine then operates like a generator and deceleration is achieved by regenerative braking. However the main disadvantage is that torque pulsations are induced at low frequencies causing speed ripple and voltage surges are impressed on the machine winding which may stress the insulation.

Invariably, in the case of a voltage source inverter drive a PWM technique is used to generate the voltage output. Recent developments in the area of microprocessors have enabled digital implementation of PWM schemes to be realised. Two basic schemes are used to operate the inverter-motor assembly: open-loop and closed-loop control schemes.

Open-loop control is the simplest form of control which essentially supplies the voltage and frequency expected to produce the correct speed and torque. A simple open-loop

inverter drive offers an important benefit. During start-up, the unit will typically supply only a low voltage and, thus, will avoid saturating the motor. It, therefore, produces more torque for a given current than a conventional direct-on-line starter. The performance of open-loop systems is, however, limited at low speeds. Torque falls when the supply frequency is below 2 Hz, and stability often becomes a problem [4].

Closed-loop control is a more sophisticated control strategy providing a range of characteristics typically accurate speed control and controlled slip for reduced energy consumption at low torque values.

A modern technique which belongs to the closed-loop control schemes is vector control. The term 'vector control' is derived from the fact that in an a.c. machine both the phase angle and the modulus of the current can be controlled. Vector control is a technique which promises high performance a.c. drives with control characteristics equal to those of the best d.c. drives. To effect satisfactory systems it is obviously necessary to control both the flux-producing and torque-producing components of current. In a d.c. motor this is possible because the field and armature currents are distinct and therefore can be individually controlled. However in the induction motor since both components of current are supplied through the three phase stator, these cannot be easily identified and controlled.

Vector control was first developed by F. Blaschke in 1972 [5]. In the mathematical model derived, equations describing the flux and torque components of the current were presented. The basis of the model is to express the voltages and currents with reference to one of three special rotating reference frames. These reference frames form the basis of all vector control schemes and consequently there are three ways of achieving vector control. The frame can be fixed either to the rotor flux phasor, stator flux phasor or the magnetising flux space phasor.

Vector control is an attractive proposition from both the technical and cost point of view, the latter due to the increasing complexity per unit cost of microelectronics. The following *Section 1.3* reviews the various aspect of vector control implementation.

1.3 Literature Review - Vector Control

The control of induction machines is fraught with complexities and technological challenge. Simple controllers exhibit non-linear characteristics and, in general, operate satisfactory only over a limited speed range. Variable frequency control involves the use of an inverter connected to the machine and until recently the design of the inverter and machine system were separate activities. However, recently the trend has been towards the integration of the design process thereby facilitating the study of the effects of the inverter output waveform on the dynamic load and vice versa. To do this effectively appropriate models of both systems are a pre-requisite. Dynamical systems require to be studied on a transient basis and consequently modelling of the induction machine using generalised machine theory is employed.

Advanced techniques, incorporating vector control have been implemented in induction machine drive systems. The objective is to develop fast acting controllers which react to load disturbances and input reference variations as fast as d.c. machine systems.

There are two ways of achieving induction machine vector control: one is direct vector control, which implies that the rotor flux vector is directly sensed by Hall probes or calculated from the stator currents and voltages. This approach is attractive in theory, yet when practically applied it attracts high cost plus possible estimation errors at low speeds.

Alternatively an indirect method can be utilised. In this method , the rotor flux vector is estimated from stator currents and the rotor speed. The major deficiency of this approach is the sensitivity to motor parameter variation caused by magnetic saturation and temperature changes.

Different approaches involving the direct method are reported in literature, for example by Kreindler, Moreira, Testa and Lipo [6]. They showed that the direct field orientation is more advantageous than the indirect type of control as it overcomes the controller sensitivity to motor parameter variations. However as direct field control requires the use of sensors, it is regarded as more difficult to apply in practice. Their work suggests the use of the third harmonic voltage component of the stator voltages to estimate the air-gap flux linkage. For a wye connection of the stator windings , the sum of the three phase voltages

is a signal dominated by the third harmonic and a high frequency component due to the rotor slot ripple and it is shown that the third harmonic can be used to estimate amplitude and position of the air-gap flux.

Generalised control schemes are proposed by several researchers. Ogasawara, Akagi and Nabae [7] propose a flux-forward control scheme that can operate for an induction machine as well as for a synchronous machine. De Doncker and Novotny [8] offer a scheme which allows a flexible change of the flux reference frame. Their conclusion is that in direct field orientation, decoupling should be done in the reference frame in which the flux is sensed, for example the reference frame fixed to the stator flux is to be used when stator flux is sensed by flux coils. The authors offer six different field orientation schemes embedded in one piece of hardware/software. Similar to the work presented in [8], where different reference frames are used and compared, Ojo, Vipin and Bhat [9] compare the steady-state performance of the induction motor drive controlled in three ways: stator field, air-gap field and rotor field oriented. They include magnetising flux saturation and conductor temperature change which influences the rotor resistance in their model.

Natural Field Orientation is analysed in [10] by Zdenkovic, Kuljic and Pasalic. Their method suggests holding the rotor flux vector modulus constant and considering the total leakage factor to be equal to zero. While the use of a speed sensor is excluded, current measurements are required. The simplicity of the control scheme allows the use of a low cost microcontroller. Ignoring the traditional ways of choosing the reference frame, Coussens, Van den Bossche and Melkebeek [11] chose to utilise a reference frame fixed to the rotor. They estimate the rotor flux using a non-linear approximation and transform the space vectors by 2 hardware vector rotation blocks from *Analog Devices*, AD2S100.

Jansen and Lorenz [12] report on two schemes for achieving rotor flux oriented control, one is direct, that is the rotor flux and angle are directly measured and an indirect method utilising the inherent slip relationship.

In direct systems the tachogenerators or digital encoders lower the system reliability and for some applications the mounting of these sensors may be difficult due to their size. As the advantage of sensorless control is evident, considerable attention has been given to the design of inexpensive estimators and therefore several tacholess vector control schemes have been studied. Peng and Fukao [13] have published work on speed sensorless control

in which the speed estimation is gained by observing the instantaneous reactive power of the rotor flux. The scheme does not require knowledge of the stator resistance or the use of an integrator as other previous MRAS schemes. Nonaka and Neba [14] measured values of motor voltages and currents in a current source inverter system, from which they derived their speed estimation. Supporters of sensorless control include Chang and Yeh [15]. They report good results employing a voltage predictor and a partial state torque current estimator, claiming that their method reduces hardware component and overall product cost.

Kim, Sul and Park [16] use an extended Kalman filter to estimate the speed. The filter is employed to identify the speed and the rotor flux based on measured quantities of stator currents and d.c. link voltage. The proposed extended Kalman filter is implemented using a 32-bit floating point DSP, TMS320C30.

In sensorless systems the magnitude and position of the rotor flux have to be known in order to decouple the flux and torque producing components of the stator current . To effect this, state observers have been used to obtain flux estimation . For example, Dalla Mora, Mares and Parasiliti [17] modified an open-loop observer into a non-linear observer which allows the non-linear dynamics of the induction motor to be accounted for. The saturation effects are taken into account by using a variable magnetising inductance within the model, the value of which is dependent upon machine supply and operating conditions.

Profumo, Griva, Pastorelli, Moreira and De Doncker [18] continued the work of De Doncker [8] and Moreira [6] and developed an air-gap flux detector based on sensing the third harmonic of voltage. It is applicable to a standard induction motor and allows the flux calculator to be simple and independent of the machine parameters. Due to the robustness of the third harmonic sensing method and to the structure of the Universal Field Oriented (*UFO*) Controller, proposed in [18], the resulting drive is not affected by detuning errors in steady state operation.

In [19] Manes, Parasiliti and Tursini described the implementation of a field-oriented control algorithm and a non-linear observer using a system based on a TMS320C40. Comparison with results given by a conventional flux estimator show excellent low sensitivity to rotor resistance variation.

Further work [20] by Manes, Parasiliti and Tursini compared results given by the non-linear observer and an extended Kalman filter. It was found that the non-linear observer exhibits the lowest numerical complexity, thus imposing less restraints when choosing the motor control hardware. Their results prove that the non-linear observer is a good solution for field-oriented induction motor control.

Wade, Dunnigan and Williams [21] presented new methods using an extended Kalman filter and Luenberger observer. As shown in previous work [22] the incorporation of a resistance representing the core losses in the machine improves the accuracy of the rotor resistance estimate. For the case of low developed torque, when difficulties in estimating the rotor resistance are encountered, they proposed the superposition of a high frequency flux controlling current on the flux current reference.

Nowadays the complexity of the field orientation method can be overcome by the use of digital control. Heinemann and Leonhard [23] use a signal processor TMS320C25 and a floating point coprocessor to perform all the calculations in field co-ordinates . They propose [23] the identification of parameters at standstill by a self-tuning algorithm. This involves an adaptive non-linear flux model which accounts for changes in temperature and iron saturation. The model tracks the saturation changes using a stored non-linear magnetising curve while the rotor resistance variation is identified by a close-loop observer.

Field-orientation systems require an accurate knowledge of the motor parameters, and of these, the most important is the rotor time constant. Wang, Novotny and Lipo [24] have proposed the use of the motor drive inverter to carry out diagnostic tests before starting the drive. They also propose to check the linear input-output torque relationship in order to verify the rotor time constant. Holtz and Thimm [25] tackled the complex problem of identifying the machine parameters by evaluating the stator current trajectory. Their technique produces good results even when operating at zero frequency or zero speed. In fact the comparison of the stator current trajectory given by an analytical machine model and the real trajectory serves as an indicator of the parameter identification scheme accuracy.

The Kalman filter is extensively used for parameter identification, for example by Atkinson, Acarnley and Finch [26]. Results for estimation of rotor current and rotor

resistance are presented. The least squares method is used by Filho, Lima and Jacobina [27] to determine the parameters of the induction machine. The identification algorithm is based on the steady state phase current versus slip and input power versus slip characteristics. The parameters are obtained as the solution of minimising the least square cost function of the difference between calculated and experimental steady state characteristics. Tungpimolrut , Peng and Fukao [28] tackle the performance degradation due to parameter variation by regulating the energy stored in the magnetising inductance. This energy can be calculated from terminal voltages and currents.

A different approach is adopted by Ganji and Lataire [29] who propose two rotor time constant tracking algorithms. The misalignment of the rotor flux vector is treated as an error function caused by having an incorrect rotor time constant. As parameter estimation is crucial to design high performance induction motor drives in [30] the authors, Ribeiro, Jacobina and Lima, use a dynamic model to determine the electrical parameters and the speed. Their model is formulated using the δ - operator. Rotor time constant estimation is also the concern of Shieh, Shyu and Liu [31] , their controller design being based on a novel dynamic estimation model of the induction motor.

Tzou, Yeh and Wu [32] deal with the rotor time constant identification and fine auto-tuning. The proposed rotor time constant identification scheme consists of two steps: firstly, for a coarse measurement of the rotor time constant, a start-up measuring process with testing phase current command at fixed slip is employed. Secondly, a fine tuning procedure is carried out under closed-loop on-line torque control. Experimental results show that the proposed algorithm can be fully automated and effectively tunes the rotor time constant to its correct value. The method implementation makes use of the calculation power of DSPs: TMS320C14 and TMS320C50. On line adaptation of the rotor time constant is also proposed in [33] (Rowan, Kerkman and Leggate). Model Reference Adaptive Control (*MRAC*) is implied by creating an error signal between a motor reference and an estimated quantity based on the motor outputs. Several reference models are proposed, for example, the torque reference model, power reference model, d and q-axis voltage reference model. Depending on the application, any of the proposed schemes may be the most suitable. On the other hand, none of them will provide a total solution to the detuning problem.

Holiday, Green, Williams [34] have developed an on-line algorithm based on introducing a low magnitude, high frequency, balanced voltage set into the motor supply. By measuring the resulting current, the combined stator and rotor resistance and inductance can be calculated. Kubota and Matsuse [35] tackle the problem of rotor resistance variation when the speed of an a.c. drive is estimated. The resulting error is eliminated by the method of simultaneously estimating the motor speed and the rotor resistance of an induction motor. Khenfer, Rezzoug, Gudefin, Meibody-Tabar [36] present a scheme which requires the motor to be driven at synchronous speed and various measurements to be taken. These are used to estimate the motor parameters before commissioning the drive.

To determine the rotor and stator resistance and the rotor time constant, Schierling [37] used different values of d.c. current applied to the motor. However, as in the work of Wang, Novotny and Lipo [24] , the commissioning is realised before the actual start of the drive. Off-line identification of motor parameters is also proposed by Ruff and Grotstollen in their work [38] in which saturation effects are considered.

Saturation in the machine obviously affects the inductive parameter values. Consequently accurate controllers require some method whereby account can be taken of these changes. Many researchers have considered this problem, for example Oh, Cho and Youn [39] propose a flux and speed controller with rotor resistance and slip frequency estimation algorithm considering the saturation effects. Herbert, Curley and Perryman [40] include both skin effects and magnetic saturation in their model. The paper describes a computer simulation of an a.c. vector controlled drive in which a finite element model of the stator and rotor assembly is used from which the motor parameters are subsequently determined. The Finite Element Method (*FEM*) is used by Lemaire-Semail, Bouillault and Razek in their work [41]. They investigated magnetic saturation and introduce a dynamic mutual inductance to take into account magnetic state variations. After calculating the parameters and their variation employing FEM , they use them in a field-oriented control where the dynamic performance heavily depends on the electrical model accuracy. Ghosh and Bhadra [42] give special attention to analysis of saturation effects. They established that the performance of a current regulated field-oriented induction motor drive degrades at high load torque if saturation is neglected. This shows that correct choice of flux level and excitation current is a vital key to designing a vector controller.

Wade, Dunnigan and Williams [22] deal with two important aspects of vector controlled induction motor drives. One is the estimation of the rotor time constant by an improved extended Kalman filter which takes into consideration the core losses. The other aspect dealt with is improving the inverter performance to extend the machine operation before the field weakening and cutting down the switching losses. They use as hardware to support their improvements implementation a DSP: Motorola 96002 , a floating point processor with a 40 MHz clock speed.

An alternative approach aimed at improving the overall performances of the system is to create efficient power sources. The output waveform of a PWM inverter can be improved by using a high ratio between the carrier frequency and the output fundamental frequency. Therefore the switching losses are increased at least in the auxiliary snubber circuit. Another aspect is regarding the ratio of fundamental voltage in the PWM output waveform to the direct supply voltage. This ratio is desired to be higher. In the conventional technique of PWM generation where a triangular carrier wave is employed, the ratio (of the fundamental to d.c. voltage) is 0.87 indicating a poor utilisation of the d.c supply. Various work tackling inverter and PWM technique improvements has been carried out. Taniguchi, Ogino and Irie [43] deal with a new PWM inverter for MOSFETs with a high carrier to fundamental frequency ratio and less heating by stopping the functioning of the inverter during one third of its period.

The current source inverters are widely used to drive an a.c. motor. In [44], Kubota, Matsue and Ree analysed a new current source GTO thyristor with an improved energy rebound scheme. Kerkman and Rowan present in [45] a voltage-controlled inverter equipped with current overload protection which provides simple start-up and operation with minimum of user interaction. It has proved to have advantages over the standard voltage inverter under overload conditions.

Pedersen and Thϕgersen [46] realised a fully-digital controlled PWM inverter having the modulation achieved via software. The modulation strategy is based on voltage vectors. Haras and Roye [47] developed a vector PWM modulator to cover the full range of inverter output voltage with minimal pulsewidth control. Though no special hardware is required for the modulator to work, general performance is limited by the microcontroller used. In the case presented the modulation frequency cannot be higher than 4 kHz. Two PWM techniques are introduced by Agelidis and Goh [48]. The inverter considered

generates a 5-level line-to-line waveform across the load. The proposed strategy results in a high performance converter with minimised filter requirements since the amplitude of all harmonics is reduced. Horstmann and Stanke [49] use a PWM modulator to calculate the switching instants. (The output voltage is regulated by a PWM technique of the constant d.c link voltage). This is suitable for application to high power drives. The controller is realised with a 16-bit microcomputer from the Intel 8086-family.

Many inverter drives are fed from a d.c. link through an input LC filter. The work reported by Walczyna, Hasse and Czarnecki [50] focus on instabilities that might occur in low pass input filters when the control is done by direct torque and flux methods. The drive is controlled by a method introduced by Takahashi (Takahashi [51] and [52]). This approach uses hysteresis flux controllers and a look-up table to keep the stator flux between two defined boundaries.

Effective vector controllers depend upon good current control. Current regulation is the subject of work by Lee, Sul and Park [53]. Current control implies two aspects: firstly, current error compensation which determines the required stator voltage and secondly, the voltage modulation which establishes the switching strategy for the inverter. The controller proposed is designed by pole placement technique utilising multivariable system regulation theory.

Many methods for current control have been reported for example: a hysteresis regulator is reported by Brod and Novotny [54], a stationary and synchronous frame proportional integral regulator by Rowan and Kerkman in [55] and Lorenz and Lawson in [56]. Ben-Brahim and Kawamura [57] propose a predictive, deadbeat regulator and Choi, Kim and Sul [58] a minimum time regulator. Chung and Sul [59] consider the current measurement error as being offset error and scaling error. Torque ripples introduced by these errors are discussed. They demonstrate that the offset error causes the torque to oscillate at the stator electrical frequency and the scaling error causes an oscillation at twice the stator frequency. Further a compensation method , which enables the speed ripple to be reduced by 66 %, is proposed. Seibel, Rowan and Kerkman [60] developed a controller for rotor-flux-oriented control to maintain current regulation and field orientation when subject to d.c. link and load disturbances while operating in the field-weakening region.

Umanand and Bhat [61] approached the design of digital-current controllers in full recognition that the vector-controlled induction motor drive is a multi-input multi-output system (MIMO). To design a good controller, that is to determine optimal controller gains, a linear combination of the quadratics of the states and the inputs needs to be minimised. This approach is called the linear quadratic (LQ). Attaianese, Damiano, Marongiu and Perfetto [62] present a prototype digital controller based on TMS320C30 DSP programmed in C language. The control algorithm is based upon the Model Reference Adaptive Control Scheme (*MRAS*) technique [63].

Particular attention to the performances at low speeds is paid by Sng, Liew [64]. They describe a MRAS used for tuning the rotor time constant or the motor speed. The parameters in the stator based equations are independently and continuously tuned using high frequency signals. The stator frame of reference is employed.

Other aspects of induction motor drive implementations are reported in literature, for example, Al-Tayie and Acarnley [65] tackle the thermal behaviour using an EKF (Extended Kalman Filter) algorithm utilising the calculation power of a DSP: TMS320C30. They are able to estimate the speed, and stator and rotor temperatures, using a simple lumped-parameter thermal model. The estimates are 'bulk' quantities, therefore this technique cannot compete with direct temperature measurement from temperature sensors. Sepe and Lang [66] discuss effects of interface components such as antialising filters. They emphasise the importance of these interfaces between the control algorithm and the drive system. Finally Chern, Liu, Jong and Yan [67] have developed a discrete-time integral variable structure model, the so-called DIVSMFC. This has proved to be fairly robust to plant parameter variation and external disturbances.

In light of the extensive activity in control applied to electrical machines good drive systems have been devised. However there is room for further improvement. The ultimate aim of the profect is to develop an inexpensive and robust control drive.

To achieve this objective, a model for the induction motor was developed. This model was realised using *DQ*-axis voltage equations for the induction motor and these equations were implemented using the Simulink toolbox within MATLAB. The transformation to *DQ*-axis for sinusoidal input conditions is described in *Chapter 2*.

In *Chapter 3* the induction motor model is explained and tested. The model is also extended to cater for general periodic input excitation waveforms. Subsequently a model for the inverter which feeds the motor is achieved. By coupling these two models together the complete drive system can be modelled and this is shown in *Chapter 4*. *Chapter 5* reviews the vector control theory.

Chapter 6 deals with motor parameter identification through simulation and practical tests. It follows the implementation of rotor-flux oriented control by means of a *Texas Instruments* Digital Motor Control (DMC) board (*Chapter 7*). Finally the testing of the vector control scheme is described and the conclusions from the work carried out and suggestions for developments are made.

Chapter 2

Theoretical approaches to induction motor modelling

2.1 Induction motor basics - Steinmetz Model

Since its discovery the induction motor has been the most widely used a.c. motor both domestically and industrially. The stator has a three-phase winding and the rotor is either *wound* or of the *squirrel-cage* type. Both types of rotor windings are short circuited during the normal operation of the machine. The machine is assumed to have an uniform air-gap and in analysis the stator and rotor slotting is neglected. The *squirrel-cage* rotor winding is constructed of copper or aluminium bars which are embedded in the rotor slots. These bars are short circuited by end rings.

Excitation is normally supplied to the stator windings which in the case of a balanced three-phase machine are displaced from each other by $2\pi/3$ electrical radians. The three-phase stator windings produce a rotating magnetic field in the air-gap and this field induces voltages and hence currents in the rotor bars.

Machine torque is produced by the interaction between the rotor currents and the air-gap field. The air-gap field rotates at a synchronous speed n_s , while the rotor, which relies upon induction effects to produce phase voltage and thereby phase currents, rotates at a steady state speed $n<n_s$.

Induction motors may be represented by electric circuit models for analysis and design synthesis purposes. For example the induction motor can be modelled by the equivalent circuit shown in *Figure 2.1.1*:

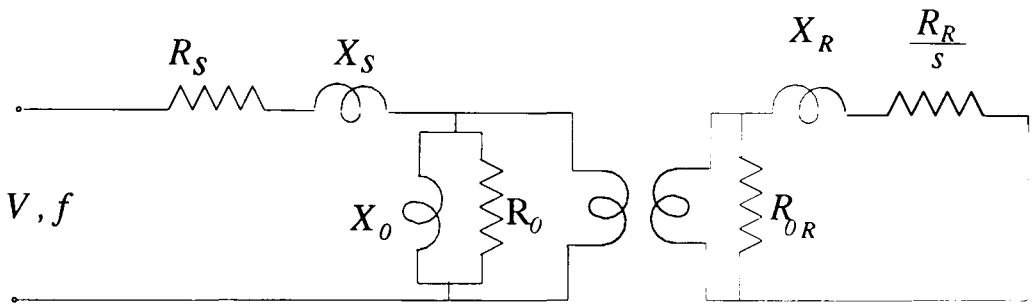

Figure 2.1.1 - Induction motor circuit

The stator and rotor may be represented as the primary and secondary windings respectively of a transformer system. In addition, because the system is balanced, a single phase may be used to represent the machine conditions.

In the circuit of *Figure 2.1.1*, V is the supply voltage, f is the supply frequency, R_S is the resistance of stator winding per phase and R_R is the rotor winding resistance per phase.

The stator and rotor leakage inductances represent the flux which does not link with both windings and can be written as: $L_S=L_{11}-L_0$ where L_S is the stator leakage inductance per phase, L_{11} is the stator winding inductance (self inductance plus mutual inductance) and L_0 is the magnetising inductance. Similarly L_R , the rotor leakage inductance per phase is derived from $L_R=L_{22}-L_0$, where L_{22} is the rotor winding inductance (self inductance plus mutual inductance).

To represent the core losses, due to eddy currents and hysteresis in the induction machine R_0 and R_{0R} are used. R_0 is the resistance which represents the stator core loss and correspondingly, R_{0R} represents the rotor core loss. Usually R_{0R} is negligible because of the low rotor induction frequency during the normal operation of the machine. This allows R_{0R} to be omitted. Moreover referring the rotor quantities to the stator circuit enable the stator and rotor circuits to be directly coupled. The resultant equivalent circuit, often known as the Steinmetz equivalent circuit, is shown in *Figure 2.1.2*:

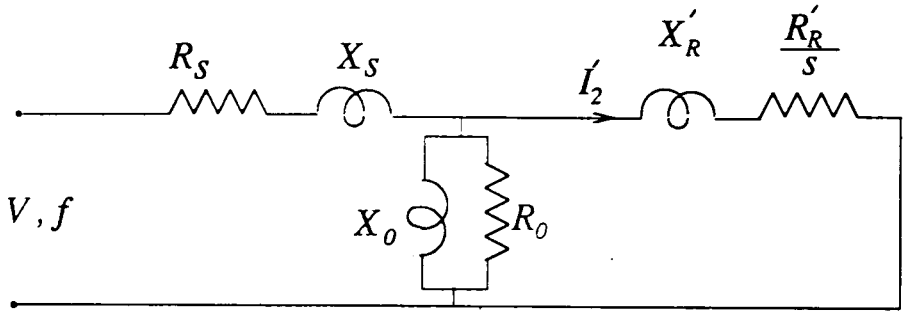

Figure 2.1.2 - Induction motor equivalent circuit

The referred rotor current I'_2 can be expressed as:

$$I'_2 = \frac{m_2}{m_1}\left(\frac{k_{w2}N_2}{k_{w1}N_1}\right)I_2 \qquad (2.1.1)$$

The notations are:

m_1 is the number of phases of stator winding,

m_2 is the number of phases of rotor winding,

k_{w1} is the winding factor of the stator winding,

k_{w2} is the winding factor of the rotor winding,

N_1 is the number of turns in stator winding and

N_2 is the number of turns in rotor winding.

The referred rotor current I'_2 represents a current flowing in N_1 turns, having m_1 phases and producing the same m.m.f. as the original I_2. When coupling the stator and rotor circuits the rotor losses I^2R must be invariant. This results in the relationship for R'_R , the referred rotor resistance, given in formula *(2.1.2)*.

$$R'_R = \frac{m_1}{m_2}\left(\frac{k_{w1}N_1}{k_{w2}N_2}\right)^2 R_R \qquad (2.1.2)$$

Similarly, X'_R the rotor reactance referred to the stator is :

$$X'_R = \frac{m_1}{m_2}\left(\frac{k_{w1}N_1}{k_{w2}N_2}\right)^2 X_R \qquad (2.1.3)$$

The input power of the circuit is noted with P_i and the power crossing the gap with P_g .

$$P_g = P_i - I_1^2 R_s - P_{core(stator)} \qquad (2.1.4)$$

where: I_1 is the stator current. All the power quantities are per phase. P_g is the power crossing the air-gap to the rotor and is represented by the power in the rotor circuit, that is:

$$P_g = I_2'^2 \frac{R_R'}{s} \qquad (2.1.5)$$

The developed mechanical power is the air-gap power minus the power loss in the rotor circuit and is therefore given by (2.1.6):

$$P_d = P_g - I_2'^2 R_r' = (1-s)P_g = \frac{1-s}{s} R_r' I_2'^2 \qquad (2.1.6)$$

The rotor resistance $\frac{R_R'}{s}$ is often rearranged into two resistances: $\frac{R_R'}{s} = R_R' + \frac{1-s}{s} R_R'$. The quantity $\frac{1-s}{s} R_R'$ is called the dynamic resistance and models the developed power. The rotor circuit power loss is modelled by R_R' .To further simplify the circuit, the stator core losses may be neglected i.e. R_0 is left out. The circuit is then as in *Figure 2.1.3*.

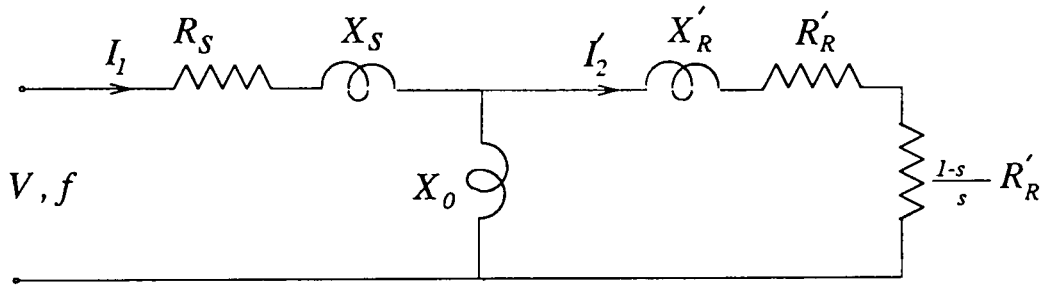

Figure 2.1.3 - Induction motor equivalent circuit

Assuming the voltage drop across R_s+jX_s to be negligible by transferring the magnetising branch to the input, another equivalent circuit, usually known as the approximate equivalent circuit is obtained as shown in *Figure 2.1.4* .

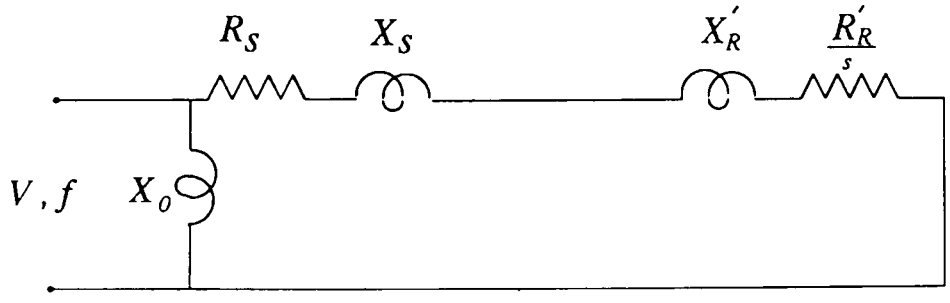

Figure 2.1.4 - Equivalent circuit

The lumped parameter circuit models developed are suitable for determining the steady-state operation of the motor under balanced conditions. However this representation does not allow transient behaviour to be modelled, therefore an alternative approach based on *DQ*-axis is adopted. The following section introduces the *DQ*-axis theory which will be then used in developing a MATLAB *DQ*-axis induction motor model.

2.2 The Three-Phase Model - Introductory Elements to DQ-axis Model

A symmetrical three-phase smooth air-gap machine with sinusoidal distributed windings is considered and the machine schematic is shown in *Figure 2.2. 1*:

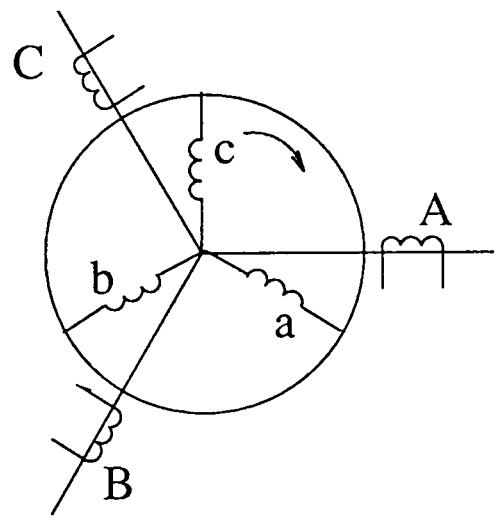

Figure 2.2. 1 - Three-phase machine

All phase-quantities are expressed in their natural reference frame. Firstly, the stator variables in the stationary reference frame fixed to the stator are described.

(2.2.1)

$$\begin{cases} v_A(t) = R_s i_A(t) + \dfrac{d\psi_A(t)}{dt} \\ v_B(t) = R_s i_B(t) + \dfrac{d\psi_B(t)}{dt} \\ v_C(t) = R_s i_C(t) + \dfrac{d\psi_C(t)}{dt} \end{cases}$$

The currents and voltages in this stationary reference frame are defined as functions of time and the instantaneous values denoted by $v_A(t), v_B(t), v_C(t)$ and $i_A(t), i_B(t), i_C(t)$ are related by *(2.2.1)* where R_s is the phase resistance of the stator windings, $\psi_A(t)$, $\psi_B(t)$, $\psi_C(t)$ are the instantaneous values of the stator flux linkages.

Similar equations can be written for the rotor and these are expressed with respect to the rotating reference frame fixed to the rotor. The rotor equations are:

$$\begin{cases} v_a(t) = R_R i_a(t) + \dfrac{d\psi_a(t)}{dt} \\ v_b(t) = R_R i_b(t) + \dfrac{d\psi_b(t)}{dt} \\ v_c(t) = R_R i_c(t) + \dfrac{d\psi_c(t)}{dt} \end{cases} \qquad (2.2.2)$$

where $v_a(t), v_b(t), v_c(t)$ and $i_a(t), i_b(t), i_c(t)$ are the instantaneous values of the rotor voltages and currents respectively, R_R is the rotor winding resistance and $\psi_a(t)$, $\psi_b(t)$, $\psi_c(t)$ are the instantaneous values of the rotor flux linkages.

The general expression for the flux linkage of a coil is:

$$\psi_1 = L_1 i_1 + \sum_{j=1}^{n} M_j i_j \qquad (2.2.3)$$

where: L_1 is the self inductance of coil *1*,

i_1 is the current flowing in coil *1*,

M_j is the mutual inductance between coil *1* and coil j,

i_j is the current which flows in coil j.

Using this expression the flux linkage of stator coil *A* can be written in the expanded form:

$$\psi_A(t) = L_A i_A(t) + M_{AB} i_B(t) + M_{AC} i_C(t) + M_{Aa} \cos\theta_R i_a(t) + M_{Ab} \cos\theta_1 i_b(t) + M_{Ac} \cos\theta_2 i_c(t) \qquad (2.2.4)$$

where: M_{AB} is the mutual inductance between the *A* and *B* stator windings,

M_{AC} is the mutual inductance between the A and C stator windings,

M_{Aa} is the mutual inductance between the A stator and a rotor windings,

M_{Ab} is the mutual inductance between the A stator and b rotor windings,

M_{Ac} is the mutual inductance between the A stator and c rotor windings.

Because of the cylindrical air-gap $M_{AB} = M_{AC} = M_S$, where M_S denotes the mutual inductance between two stator windings. Also $M_{Aa} = M_{Ab} = M_{Ac} = M_{SR}$, where M_{SR} is the mutual inductance between the stator and rotor windings. The rotor angle is denoted by θ_R. The angle between A and b phases is θ_1, defined by $\theta_1 = \theta_R + \frac{2\pi}{3}$ and the angle between A and c phases is θ_2 defined by $\theta_2 = \theta_R + \frac{4\pi}{3}$. L_A is the stator winding self inductance.

In equation *(2.2.1)* the flux linkage may be replaced by the form shown in *(2.2.4)*. Linear magnetic conditions are considered therefore, the inductances are taken as invariant quantities.

The v_A voltage equation in its expanded form is shown in *(2.2.5)*.

$$v_A = R_S i_A + L_A \frac{di_A}{dt} + M_S \frac{di_B}{dt} + M_S \frac{di_C}{dt} + M_{sr} \cos\theta_r \frac{di_a}{dt} + M_{sr} \cos\theta_1 \frac{di_b}{dt} + M_{sr} \cos\theta_2 \frac{di_c}{dt} \qquad (2.2.5)$$

Similar equations can be written for the other stator and rotor voltages to give a *6x6* non-linear matrix equation. Even when all machine parameters are considered constant, the voltage equations still contain variable coefficients. Additionally, the rotor angle may be considered to vary in time [68].

Significant simplification in the voltage equation system can be made by using the two-phase equivalent quantities in the same natural reference frames. Further reduction of the system elements is achieved by expressing the rotor voltages and currents in the stationary frame fixed to the stator. This introduces the *DQ*-axis theory presented in the following sections.

Some simplifying assumptions are made: there is no magnetic saturation, air-gap magneto-motor-forces and fluxes are represented by the fundamental components, slotting effects are ignored and eddy current and hysteresis losses are not considered.

Two mathematical stages are necessary to transform the voltage equations to the primitive *DQ* equations. These are shown schematically in *Figure 2.2.2 - Figure 2.2.4.*

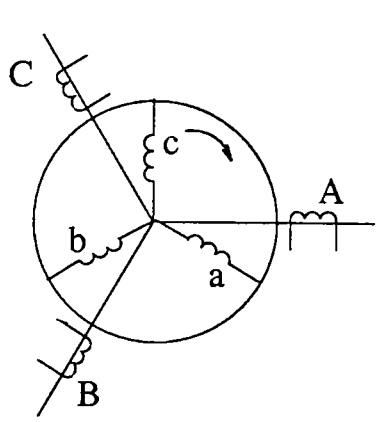

Figure 2.2.2 Three-phase

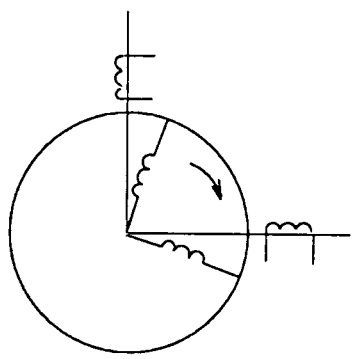

Figure 2.2.3 Three to two phase transformation

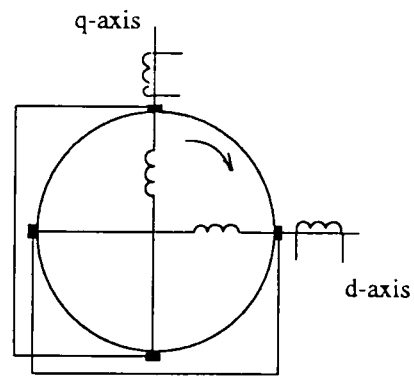

Figure 2.2.4 Commutator transformation

Initially, the 3-to-2 phase transformation, which reduces the three-phase rotor and stator quantities to two-phase values is applied, followed by the commutator transformation which transforms the rotor quantities from a two-axis rotating space-frame to a two-axis fixed space-frame. The effect of applying the second transformation is to replace the rotating winding m.m.f distribution with a stationary winding m.m.f distribution. The transformation results in rotor currents being transformed from slip to supply frequency.

The mathematical basis of these transformations are described in the next sections. The 3-to-2 phase transformation and the commutator transformation are described in *Section 2.3* and *Section 2.4* respectively. In *Section 2.5* these transformations are applied to the induction motor to obtain a *DQ* model.

2.3 The 3-phase (abc) to 2-phase ($\alpha\beta\gamma$) winding transformation

A 2-pole, balanced, symmetrical three-phase winding can be represented by three concentrated coils of N_1 turns each and displaced $120°$ as shown in *Figure 2.3.1*. The two-axis $(\alpha\beta)$ are fixed in an arbitrary position and for ease of algebraic manipulations the phase *a* axis is assumed to be coincident with α -axis.

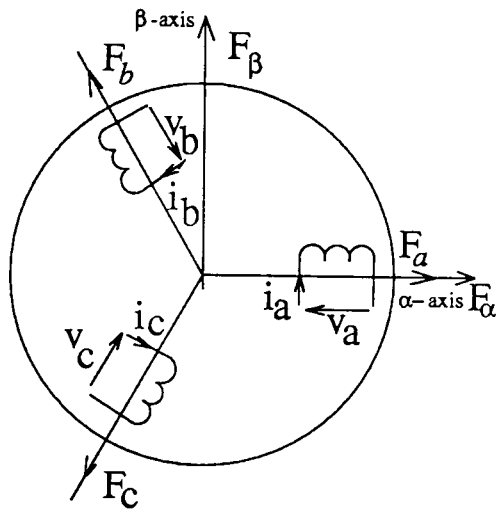

Figure 2.3.1- $(\alpha\beta\gamma)$ axes

Resolving the m.m.f.s. along the $(\alpha\beta)$ axes gives:

$$\begin{bmatrix} F_\alpha \\ F_\beta \end{bmatrix} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} F_a \\ F_b \\ F_c \end{bmatrix} \qquad (2.3.1)$$

Equation *(2.3.1)* cannot be used directly to derive the inverse relationship, because the transformation matrix is not square but a *2x3* matrix. A further equation must therefore be included to emulate the *3rd*-sequence component. The resultant matrix is given by:

$$\begin{bmatrix} F_\alpha \\ F_\beta \\ F_\gamma \end{bmatrix} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ m & m & m \end{bmatrix} \begin{bmatrix} F_a \\ F_b \\ F_c \end{bmatrix} \qquad (2.3.2)$$

where $\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ m & m & m \end{bmatrix} = [A]$

By arbitrary choosing $m=\frac{1}{\sqrt{2}}$ the inversion of the matrix is a simple process of transposition. The matrix [B] is introduced by equations *(2.3.3)* and *(2.3.4)*.

$$[A]=\sqrt{\frac{3}{2}}\sqrt{\frac{2}{3}}\begin{bmatrix}1 & -\frac{1}{2} & -\frac{1}{2}\\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2}\\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix}=\sqrt{\frac{3}{2}}[B] \qquad (2.3.3)$$

$$[A^{-1}]=\sqrt{\frac{2}{3}}\sqrt{\frac{2}{3}}\begin{bmatrix}1 & 0 & \frac{1}{\sqrt{2}}\\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & \frac{1}{\sqrt{2}}\\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & \frac{1}{\sqrt{2}}\end{bmatrix}=\sqrt{\frac{2}{3}}[B_t] \qquad (2.3.4)$$

and therefore $[B^{-1}]=[B_t]$.

The transformation between the *(abc)* three-phase windings to *(αβ)* two-phase equivalent windings is shown in *(2.3.5)*. The corresponding inverse relationship is given by *(2.3.6)*.

$$[F_{\alpha\beta\gamma}]=\sqrt{\frac{3}{2}}[B][F_{abc}] \qquad (2.3.5)$$

$$[F_{abc}]=\sqrt{\frac{2}{3}}[B_t][F_{\alpha\beta\gamma}] \qquad (2.3.6)$$

If the αβ and *abc* coils are assumed to have N_2 and N_1 turns respectively, then the m.m.f.s. can be written as:

$$[F_{\alpha\beta\gamma}]=N_2[i_{\alpha\beta\gamma}] \qquad (2.3.7)$$
$$[F_{abc}]=N_1[i_{abc}]$$

Further, the transformation between the two frames for currents is worked out as being:

$[i_{\alpha\beta\gamma}]=\frac{N_1}{N_2}[i_{abc}]\frac{[F_{\alpha\beta\gamma}]}{[F_{abc}]}=\frac{N_1}{N_2}[i_{abc}]\sqrt{\frac{3}{2}}[B]$, whereby choosing $\frac{N_1}{N_2}=\sqrt{\frac{2}{3}}$ the relation can be further simplified. In *(2.3.8)* current transformation relationships are presented.

$$[i_{\alpha\beta\gamma}]=[B][i_{abc}] \qquad (2.3.8)$$
$$[i_{abc}]=[B_t][i_{\alpha\beta\gamma}]$$

The primitive machine has to be power invariant with respect to the original machine. The power relationships yield the corresponding transformation for voltage in *(2.3.9)*.

$$[v_{\alpha\beta\gamma}]=[B][v_{abc}] \qquad (2.3.9)$$
$$[v_{abc}]=[B_t][v_{\alpha\beta\gamma}]$$

The voltage/current relationship for both coil systems can be written as:

$$[v_{abc}]=[Z_{abc}][i_{abc}] \qquad (2.3.10)$$

$$[v_{\alpha\beta\gamma}]=[Z_{\alpha\beta\gamma}][i_{\alpha\beta\gamma}] \qquad (2.3.11)$$

where $[Z_{abc}]$ is the impedance matrix and is given by:

$$[Z_{abc}]=\begin{bmatrix} R_a & 0 & 0 \\ 0 & R_b & 0 \\ 0 & 0 & R_c \end{bmatrix}+\begin{bmatrix} L_a & M_{ab} & M_{ac} \\ M_{ab} & L_b & M_{bc} \\ M_{ac} & M_{bc} & L_c \end{bmatrix}p=[R_{abc}]+[L_{abc}]p \qquad (2.3.12)$$

where $[R_{abc}]=\begin{bmatrix} R_a & 0 & 0 \\ 0 & R_b & 0 \\ 0 & 0 & R_c \end{bmatrix}$ and $[L_{abc}]=\begin{bmatrix} L_a & M_{ab} & M_{ac} \\ M_{ab} & L_b & M_{bc} \\ M_{ac} & M_{bc} & L_c \end{bmatrix}$ *(2.3.13)*

In the previous equations the notations are:

R_a, R_b and R_c are per phase *(abc)* winding resistances,

L_a, L_b and L_c are per phase *(abc)* winding self inductances,

$M_{ac}=M_{ca}$, $M_{ab}=M_{ba}$, $M_{bc}=M_{cb}$ are the mutual inductance between stator phases.

In *(2.3.10)* $[i_{abc}]$ is replaced by its expression of *(2.3.8)* and $[v_{abc}]$ with *(2.3.9)* and then both sides are multiplied by [B]. By comparing the result with *(2.3.11)*, the impedance matrix $[Z_{\alpha\beta\gamma}]$ is found to be:

$$\left[Z_{\alpha\beta\gamma}\right] = [B][Z_{abc}][B_t] = [B][R_{abc}][B_t] + [B][L_{abc}][B_t]p = \left[R_{\alpha\beta\gamma}\right] + \left[L_{\alpha\beta\gamma}\right]p \qquad (2.3.14)$$

The equations are valid also for unsymmetrical windings. In case of a symmetrical three-phase winding in an uniform air-gap machine, per phase quantities and mutual coefficients are equal for all three phases as shown in *(2.3.15)*.

$$R_a = R_b = R_c = R \qquad L_a = L_b = L_c = L \qquad M_{ab} = M_{bc} = M_{ac} = M \qquad (2.3.15)$$

Applying *(2.3.15)*, the resistance and inductance transformations are given by *(2.3.16)*.

$$\left[R_{\alpha\beta\gamma}\right] = \left[R_{abc}\right] = \begin{bmatrix} R & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & R \end{bmatrix} \text{ and } \left[L_{\alpha\beta\gamma}\right] = \begin{bmatrix} (L-M) & 0 & 0 \\ 0 & (L-M) & 0 \\ 0 & 0 & (L+2M) \end{bmatrix} \qquad (2.3.16)$$

From the general flux linkage relationship $[\psi] = [L][i]$ and using *(2.3.8)* and *(2.3.14)*, the following flux transformations are worked out:

$$\left[\psi_{\alpha\beta\gamma}\right] = [B][\psi_{abc}] \qquad (2.3.17)$$

$$[\psi_{abc}] = [B_t]\left[\psi_{\alpha\beta\gamma}\right]$$

2.4 The rotating 3-phase *(abc)* to stationary 2-phase *(dq)* windings transformation

Whilst only the 3-to-2 phase winding transformation is required for stator parameters, as described in *Section 2.3*, rotor quantities need a further transformation to fix the rotating windings to a stationary frame. As with the stator parameters, the 3-to-2 phase transformation is carried out but in the case of the rotor the equivalent $(\alpha\beta)$ axes rotate at an angular speed ω_r. If the angular displacement between the two sets of axes, shown in *Figure 2.4.1*, is given by an angle θ_r, F_α and F_β can be resolved along the stationary *(dq)* axes and are given by:

$$\begin{bmatrix} F_d \\ F_q \end{bmatrix} = \begin{bmatrix} \cos\theta_r & \sin\theta_r \\ -\sin\theta_r & \cos\theta_r \end{bmatrix} \begin{bmatrix} F_\alpha \\ F_\beta \end{bmatrix} \qquad (2.4.1)$$

With the condition of zero contribution to the space resultant of F_α and F_β , F_γ is included in *(2.4.1)* to give a *3x3* matrix. This will facilitate combination with equation *(2.3.5)* thus yielding a relationship between *(abc)* and $(dq\gamma)$.

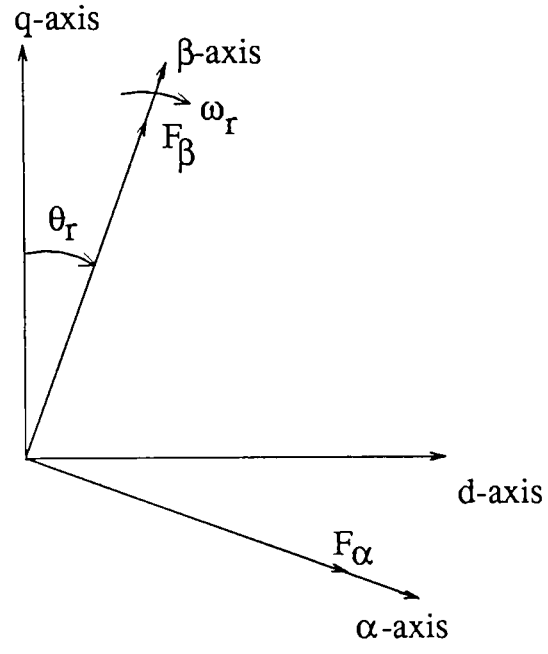

Figure 2.4.1 - the orthogonal dq-axes

$$\begin{bmatrix} F_d \\ F_q \\ F_\gamma \end{bmatrix} = [S] \begin{bmatrix} F_\alpha \\ F_\beta \\ F_\gamma \end{bmatrix} \qquad \text{where } [S] = \begin{bmatrix} \cos\theta_r & \sin\theta_r & 0 \\ -\sin\theta_r & \cos\theta_r & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (2.4.2)$$

The inverse of [S] is actually its transpose i.e., $[S^{-1}] = [S_t]$, thus allowing *(2.4.3)* to be written in a condensed form as:

$$[F_{dq\gamma}] = [S][F_{\alpha\beta\gamma}] \qquad (2.4.3)$$
$$[F_{\alpha\beta\gamma}] = [S_t][F_{dq\gamma}]$$

Using *(2.3.5)*, $[F_{\alpha\beta\gamma}] = \sqrt{\frac{3}{2}}[B][F_{abc}]$ the transformation from *(abc)* to *(dq)* is given by *(2.4.4)*.

$$[F_{dq\gamma}] = \sqrt{\frac{3}{2}}[C][F_{abc}] \qquad \text{where: } [C] = [S][B] \qquad (2.4.4)$$

Finally, it is possible to transform directly from *(abc)* to *(dq)* frame using the [C] matrix. The reverse transformation, from *(dq)* to *(abc)* uses the inverse of [C], $\left[C^{-1}\right]=\left[C_t\right]$.

$$[C]=\sqrt{\frac{2}{3}}\begin{bmatrix}\cos\theta_r & \cos(\theta_r-120) & \cos(\theta_r+120)\\ -\sin\theta_r & -\sin(\theta_r-120) & -\sin(\theta_r+120)\\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix} \qquad (2.4.5)$$

Transformation relationships between the *(abc)* axes and the *(dq)* axes are summarised in *(2.4.6)*.

$$\begin{aligned}\left[i_{dq\gamma}\right]&=[C][i_{abc}]\\ \left[\psi_{dq\gamma}\right]&=[C][\psi_{abc}]\\ \left[v_{dq\gamma}\right]&=[C][v_{abc}]\\ \left[L_{dq\gamma}\right]&=[C][L_{abc}][C_t]\\ \left[Z_{dq\gamma}\right]&=[C][Z_{abc}][C_t]\end{aligned} \qquad (2.4.6)$$

2.5 The primitive machine

The transformation of the induction machine to an equivalent primitive machine is realised in two stages: the first step involves the transformation of the three-phase voltages and currents to their two-phase equivalent. Since the stator is stationary the $(\alpha\beta)$ system can be made to coincide with the *(dq)* system. However the rotor quantities require a further transformation stage, i.e. from the rotating two-axis rotor space-frame to the stationary two-axis reference space-frame of the stator.

Transformations for both the stator and rotor parameters have been discussed in detail in the previous sections and will be now applied to the modelling of an induction motor.

Using equation *(2.3.8)* the stator currents, expressed in *DQ*-axis frame, are given by:

$$\left[i_{DQ\Gamma}\right]=[B]\left[i_{ABC}\right] \Rightarrow \begin{bmatrix} i_D \\ i_Q \\ i_\Gamma \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix} \qquad (2.5.1)$$

where subscripts D, Q refer to the stator and d,q to the rotor quantities. The transformation from *(abc)* to $(\alpha\beta)$ is applied to rotor currents in *(2.5.2)*, complying with *(2.3.8)*.

$$\begin{bmatrix} i_\alpha \\ i_\beta \\ i_\gamma \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \qquad (2.5.2)$$

Then the transformation from $(\alpha\beta\gamma)$ to $(dq\gamma)$ is performed :

$$\begin{bmatrix} i_d \\ i_q \\ i_\gamma \end{bmatrix} = \begin{bmatrix} \cos\omega_r t & \sin\omega_r t & 0 \\ -\sin\omega_r t & \cos\omega_r t & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \\ i_\gamma \end{bmatrix} \qquad (2.5.3)$$

The direct transformation from *(abc)* to $(dq\gamma)$ can be obtained by applying *(2.5.4)*.

$$\left[i_{dq\gamma}\right]=[S][B]\left[i_{abc}\right]=[C]\left[i_{abc}\right] \qquad (2.5.4)$$

Transformation relationships for stator quantities are given by:

$$\left[i_{DQ}\right]=[B]\left[i_{ABC}\right] \qquad (2.5.5)$$

$$\left[v_{DQ}\right]=[B]\left[v_{ABC}\right] \qquad (2.5.6)$$

and for rotor parameters these are:

$$\left[i_{dq}\right]=[C]\left[i_{abc}\right] \qquad (2.5.7)$$

$$\left[v_{dq}\right]=[C]\left[v_{abc}\right] \qquad (2.5.8)$$

For a conventional three-phase induction machine the winding transformations give a four coil primitive machine, shown in *Figure 2.5.1* and the application of equations *(2.5.5)-(2.5.8)* to this model yields:

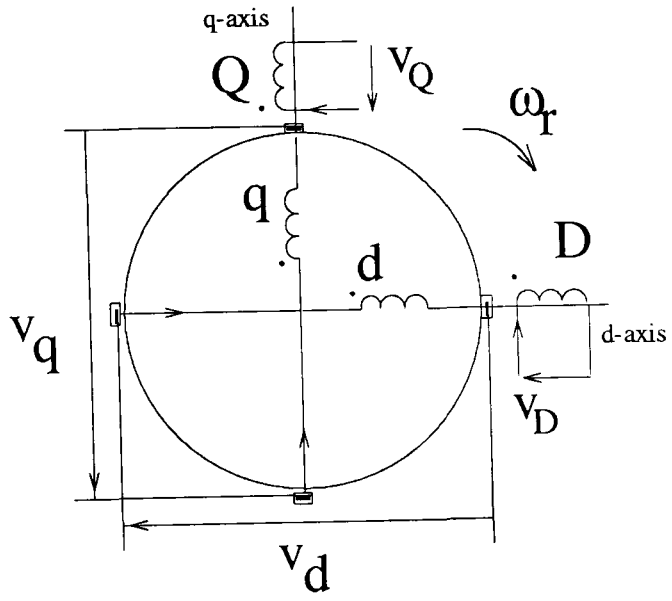

Figure 2.5.1 - The primitive machine

$$\begin{cases} v_D = R_D i_D + L_D \dfrac{di_D}{dt} + M_{Dd} \dfrac{di_d}{dt} \\ v_Q = R_Q i_Q + L_Q \dfrac{di_Q}{dt} + M_{Qq} \dfrac{di_q}{dt} \\ v_d = R_d i_d + L_d \dfrac{di_d}{dt} + M_{dD} \dfrac{di_D}{dt} + \omega_r G_{dQ} i_Q + \omega_r G_{dq} i_q \\ v_q = R_q i_q + L_q \dfrac{di_q}{dt} + M_{qQ} \dfrac{di_Q}{dt} + \omega_r G_{qD} i_D + \omega_r G_{qd} i_d \end{cases} \qquad (2.5.9)$$

where:

$R_s = R_D = R_Q$ is the stator phase resistance,

$R_r = R_d = R_q$ is rotor phase resistance,

$M_{Dd} = M_{dD} = M_{Qq} = M_{qQ} = M = L_m$ is the mutual inductance,

$L_D = L_Q$ is the stator self inductance,

$L_d = L_q$ is the rotor self inductance,

$G_{qD} = -G_{dQ} = M = L_m$ and $G_{qd} = -G_{dq} = L_d$ are the rotational inductance coefficients.

If a balanced three-phase set of voltages is defined as:

$$\begin{bmatrix} v_A \\ v_B \\ v_C \end{bmatrix} = V \begin{bmatrix} \cos\ \omega t \\ \cos(\ \omega t - 120^\circ) \\ \cos(\ \omega t - 240^\circ) \end{bmatrix} \qquad (2.5.10)$$

then the DQ-axis voltages are given by *(2.5.11)*:

$$v_D = \sqrt{\frac{3}{2}} v_A = \sqrt{\frac{3}{2}} V \cos\ \omega t \qquad (2.5.11)$$

and

$$v_Q = -\sqrt{\frac{3}{2}} V \sin\ \omega t = \sqrt{\frac{3}{2}} V \cos(\ \omega t + 90^\circ)$$

The electromagnetic torque is derived from the energy equation as shown in *(2.5.12)*.

$$T_e = \frac{pp}{\omega_r} \frac{dW_{em}}{dt} \Rightarrow \quad T_e = pp[i_t][G][i] \qquad (2.5.12)$$

$$\text{where } [G] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -M & 0 & -L_d \\ M & 0 & L_d & 0 \end{bmatrix} \qquad (2.5.13)$$

pp is the number of pole pairs and $[G]$ is the rotational inductance coefficients matrix. Thus the expression for the torque is given by:

$$T_e = (pp) M (i_q i_D - i_d i_Q) \qquad (2.5.14)$$

To fully define the performance of the induction motor the system motion equation needs to be integrated with the DQ-axis equations. The load torque of the motor may be due to friction, windage, acceleration and mechanical work. [69]. By neglecting the coulomb and static friction components, the torque due to friction is approximated to: $T_F = B\omega_r$, called viscous friction, where B is a constant for the system and ω_r is the rotor speed. The windage torque is combined with the viscous torque and is given by: $T_{F+W} = D\omega_r$ where D, the damping constant, is appropriately chosen. The component of the torque required to accelerate the system is expressed as $T_J = J \frac{d\omega_r}{dt}$, where J is the rotational inertia of the

system in [kg·m^2]. Finally, the last component is noted with T_L, the load torque, given by the mechanical work required. It can be expressed as a function of ω_r . Therefore, the developed electromagnetic torque T is given by:

$$T = J\frac{d\omega_r}{dt} + D\omega_r + T_l \qquad (2.5.15)$$

The rotor speed, ω_r can be expressed as $\omega_r = \frac{d\theta_r}{dt}$, where θ_r is the electrical rotor angle. The electrical rotor angle can be further expressed as $\theta_r = pp\theta_{rm}$ where θ_{rm} is the mechanical rotor angle.

It can be shown that for steady state operation equations *(2.5.9)* reduce to a system of equations which describes the conventional Steinmetz circuit. However, for transients no simplification exists. Fortunately, sophisticated software packages are available for modelling and solving the transient equations. In this research MATLAB and Simulink toolbox are utilised for this purpose. In the following chapter a suitable induction motor model is developed using the Simulink toolbox and various simulations are carried out for a particular set of motor data to test its validity.

Chapter 3

DQ-axis induction motor modelling

The objective of this chapter is to present a model developed for an induction motor using the *DQ*-axis theory previously introduced. The model proposed has been designed using the Simulink toolbox within the MATLAB software package. Therefore the first section tries to familiarise the reader with MATLAB/ Simulink by shortly mentioning its usage and a few features of the software.

The induction motor model is then introduced together with its Simulink block diagram. To prove the validity of the model, various simulations are carried out, the corresponding results plotted and a few commentaries given.

3.1 Simulink Toolbox within MATLAB

MATLAB, the acronym for Matrix Laboratory is an integrated technical computing software package that combines numeric computation, advanced graphics and visualisation, with a high-level programming language.

It is arguably the leader in this area of software systems modelling with applications ranging from signal and image processing to medical research. MATLAB also features a family of application-specific toolboxes which are comprehensive collections of functions that extend the MATLAB environment to solve particular classes of problems. One of the toolboxes provided by MATLAB is the Simulink Toolbox [70].

Simulink provides an interactive environment for modelling, analysing and simulating a wide variety of dynamic systems. Simulink supplies a graphical user interface for constructing block diagram models. By simple `drag and drop` operations different components from the Simulink's block library can be defined and connected using a mouse to create rapidly the model of a system. Hierarchical models can be developed by grouping blocks into subsystems. Simulations can be run from both pull-down menus or in batch mode from the command line. Results can be displayed during simulations using scope and graph blocks functions [71].

3.2 Modelling the induction motor

Using the voltage equations given in *(2.5.9)* and the torque equations given in *(2.5.14)* and *(2.5.15)* a model for the induction motor was developed using the Simulink toolbox within MATLAB software. The first voltage equation:

$$v_D = R_D i_D + L_D \frac{di_D}{dt} + M_{Dd} \frac{di_d}{dt} \qquad (3.2.1)$$

can be rewritten as:

$$L_D \frac{di_D}{dt} = v_D - R_D i_D - M_{Dd} \frac{di_d}{dt} \qquad (3.2.2)$$

so that the component associated with the highest order differential can be assumed to be available. In this case circuit D is under consideration and $\frac{di_D}{dt}$ is the highest order differential. The remaining components of the equation are derived from the other primitive circuits. A typical `*patch*` diagram for v_D equation is shown in *Figure 3.2.1.*

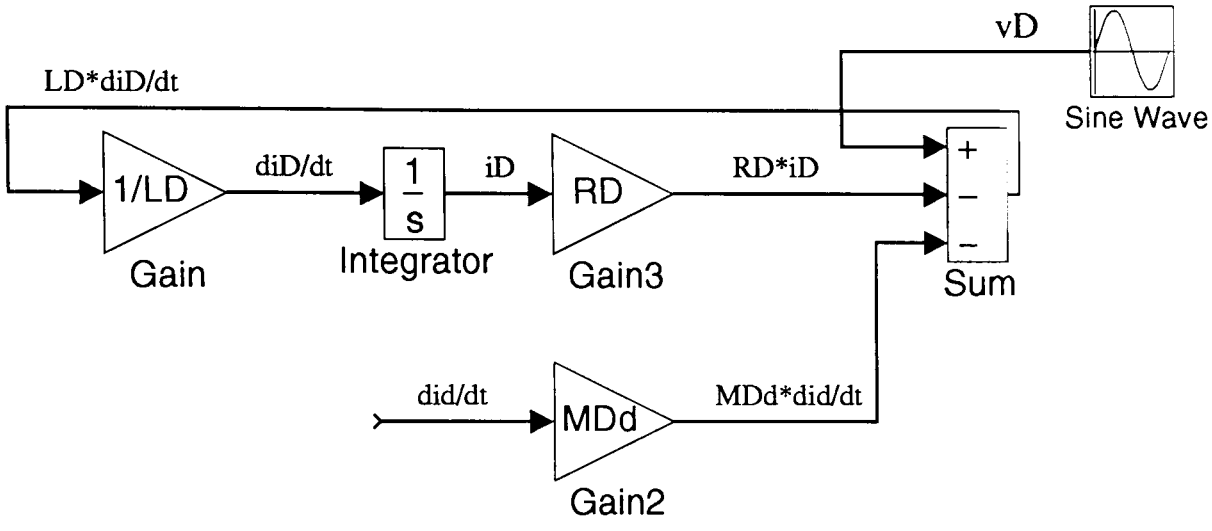

Figure 3.2.1 - The Simulink block diagram for modelling the first equation

All other equations can be patched and coupled together in a similar manner to give the overall machine model as shown in *Figure 3.2.2*.

In order to test the validity of the motor model, simulations are carried out. For this purpose parameters for a *415 V*, *3-phase*, *4-pole*, *4 kW*, *delta*-connected cage induction motor were used. The motor has been well tested [72] and found to have a rated current of *8 A* and friction and windage losses of *20 W* at operating speed.

The motor parameters are:

$R_1 = R_s = 5\Omega$ is the phase stator resistance $\Rightarrow R_D = R_Q = 5\Omega$,

$X_{L_1} = 8.66\Omega$ is the phase stator leakage reactance,

$R_2 = R_r = 3.52\Omega$ is the phase rotor resistance $\Rightarrow R_d = R_q = 3.52\Omega$,

$X_{L_2} = 8.66\Omega$ is the phase rotor leakage reactance,

$X_{L_m} = 170\Omega$ is the magnetising reactance and

$R_m = 1848\Omega$ is the stator core loss resistance.

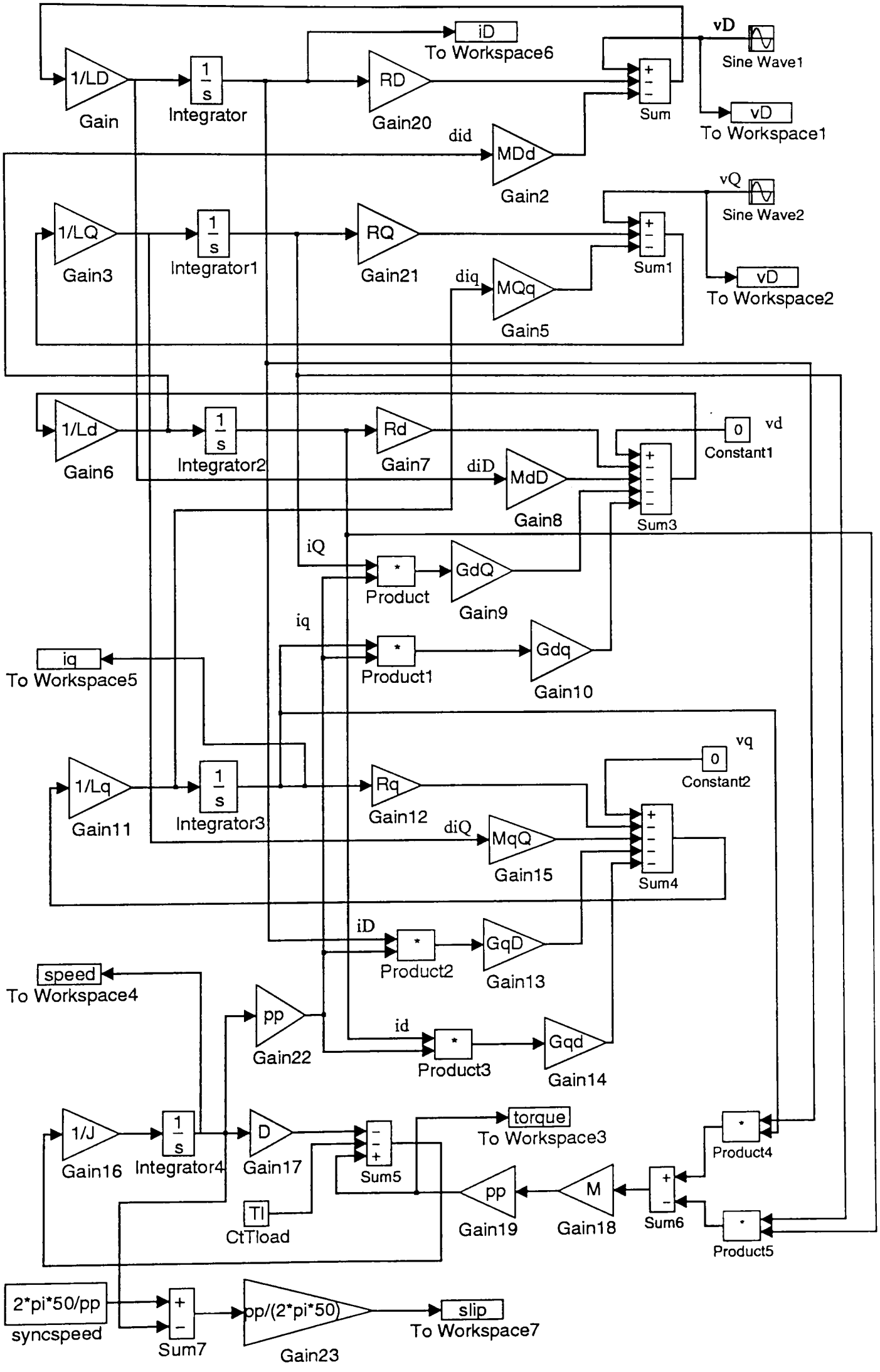

Figure 3.2.2 - The Simulink block diagram for the induction motor model

From these values the magnetising inductance, M and the stator leakage inductance can be calculated, for example:

$$M = L_m = \frac{X_{L_m}}{2\pi f} = \frac{170}{2\pi 50} = 0.5411 \text{ and } L_1 = \frac{X_{L_1}}{2\pi f} = \frac{8.66}{2\pi 50} = 0.0276 .$$

The DQ-axis stator voltages are given by equation *(2.5.11)* as :

$$v_D = \sqrt{\frac{3}{2}}\, V\cos\omega t = \sqrt{\frac{3}{2}} 415 \cos\omega t = 508.27 \cos\omega t,$$

$$v_Q = -\sqrt{\frac{3}{2}}\, V\sin\omega t = -508.27 \sin\omega t .$$

The stator and rotor inductances in the DQ frame, L_d and L_D are approximated to:

$$L_d = \frac{X_2}{\omega} + M = 0.0276 + 0.5411 = 0.5687 , \; L_q = L_d = 0.5687 .$$

$$L_D \cong \frac{X_1}{\omega} + M = 0.5687 = L_Q \quad \text{using the approach suggested by O'Kelly [73].}$$

The data file for running the simulation is given in Appendix 1. The parameters of the chosen motor define the Simulink `*Gain*` and `*Sinewave*` blocks.

In *Figure 3.2.2* can also be noticed `*To workspace*` blocks. The signal entering a block of this type is stored in a matrix (vector) in the workspace at the end of the simulation. This allows various graphs to be plotted when simulation is completed.

A simulation is defined by parameters like: the method used (*Runge-Kutta 5*), start and stop times (*0 and 0.6 seconds*) and minimum and maximum integration step sizes (*0.0001 and 0.001*).

As all Simulink blocks and simulation parameters were defined, simulations were carried out for two different values of inertia, $J=0.005$ and $J=0.01$ over a period of *0.6 seconds* and the results are plotted on the same graphs to enable better comparison.

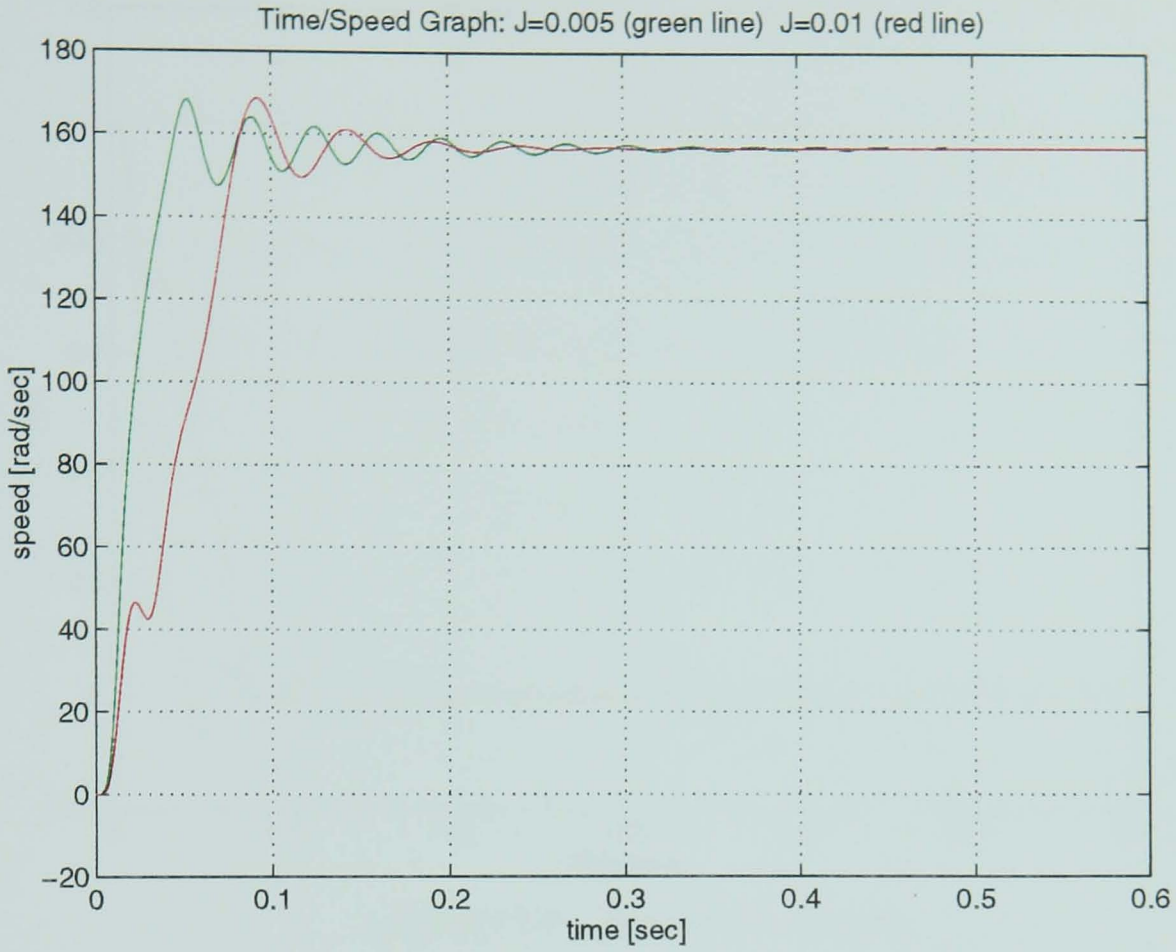

Figure 3.2.3 - Speed plotted versus time

In *Figure 3.2.3*, where speed is plotted against time the oscillatory transient behaviour of the machine can be clearly seen.

Comparing the two speed/time plots corresponding to the two different values of inertia shows that the motor model behaves as expected. When *J=0.01*, the higher inertia causes the motor to reach the oscillation period after a delayed time with respect to the speed graph of *J=0.005*.

When *J=0.005*, the synchronous speed is $\omega_s = \frac{2\pi fq}{pp} = \frac{2\pi(50)}{2} = 157.08\ rad/s$, the simulated steady-state rotor speed is ω_r=*157.0325 rad/s* and slip is $s = \frac{\omega_s - \omega_r}{\omega_s}$=*0.0003*.

The plot of slip against time is shown in *Figure 3.2.4*.

It can be seen that at starting the value for slip is *s=1*, which corresponds to ω_r=*0 rad/s* and then the rotor goes through a transient interval before settling at a steady state slip of *0.0003*.

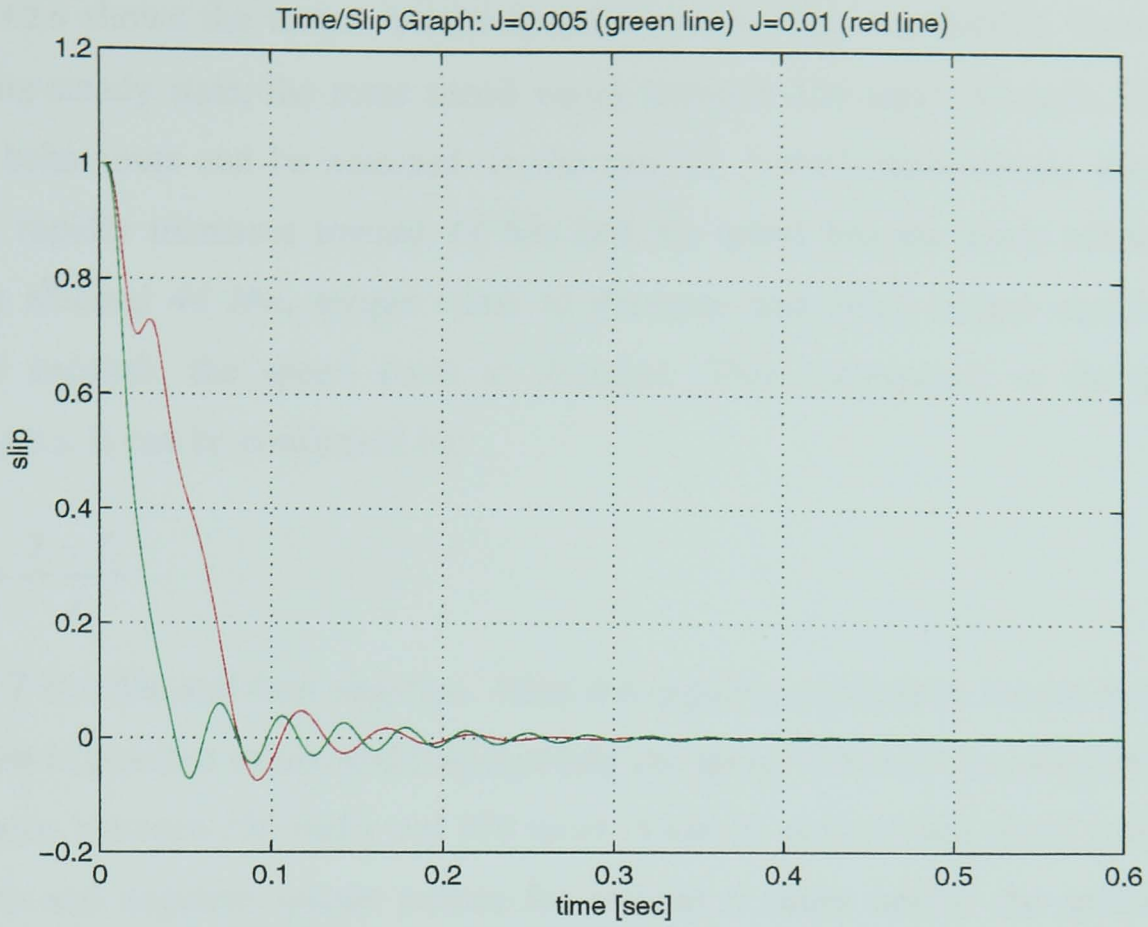

Figure 3.2.4 - Slip plotted versus time

Figure 3.2.5 shows the variation of torque with time. For $J=0.01$ the peak transient torque is, at starting, about *44 Nm* and at steady state, after the oscillations have decayed the steady state value is *0.1 Nm*. This is the value at which the load torque has been pre-specified since at steady state $d\omega_r/dt \rightarrow 0$ and damping constant, D is zero [Ref: *(2.5.15)*].

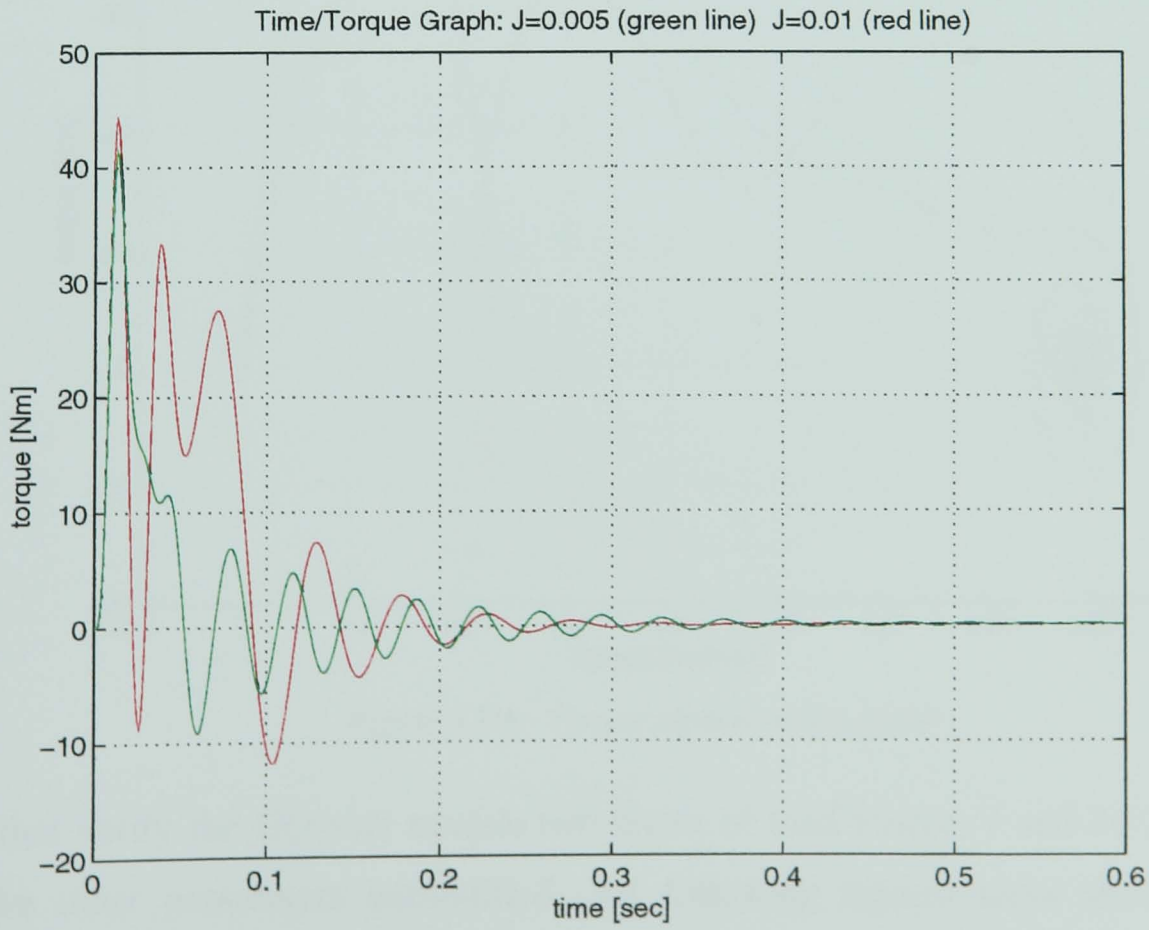

Figure 3.2.5 - Torque plotted versus time

Figure 3.2.6 shows the torque versus speed response. The oscillations show that prior to reaching steady state, the rotor speed varies between 150 and 170 *rad/s* .The torque and speed behaviours can be assessed. In the case of *J=0.01*, immediately after starting the torque rapidly increases toward *44 Nm* and the speed has the same conduct. However, having reached *44 Nm*, torque starts to decrease and when torque equals *0 Nm*, after *0.0231 seconds*, the speed starts to decrease. This corresponds to the glitch seen in *Figure 3.2.3*. It can be confirmed by:

$$\frac{d\omega_r}{dt} = \frac{T - T_l}{J} \qquad (3.2.3)$$

where T is *0 Nm* and then negative. After the negative oscillation, torque recovers to *0 Nm* and then to positive values and consequently the speed increases. Looking in *Figure 3.2.6*, at the region between *150 rad/s* and *170 rad/s*, it can be noticed that speed oscillation caused by zero and negative torque occurs for another 5 times before the speed achieves the steady-state value of *157.0426 rad/s*.

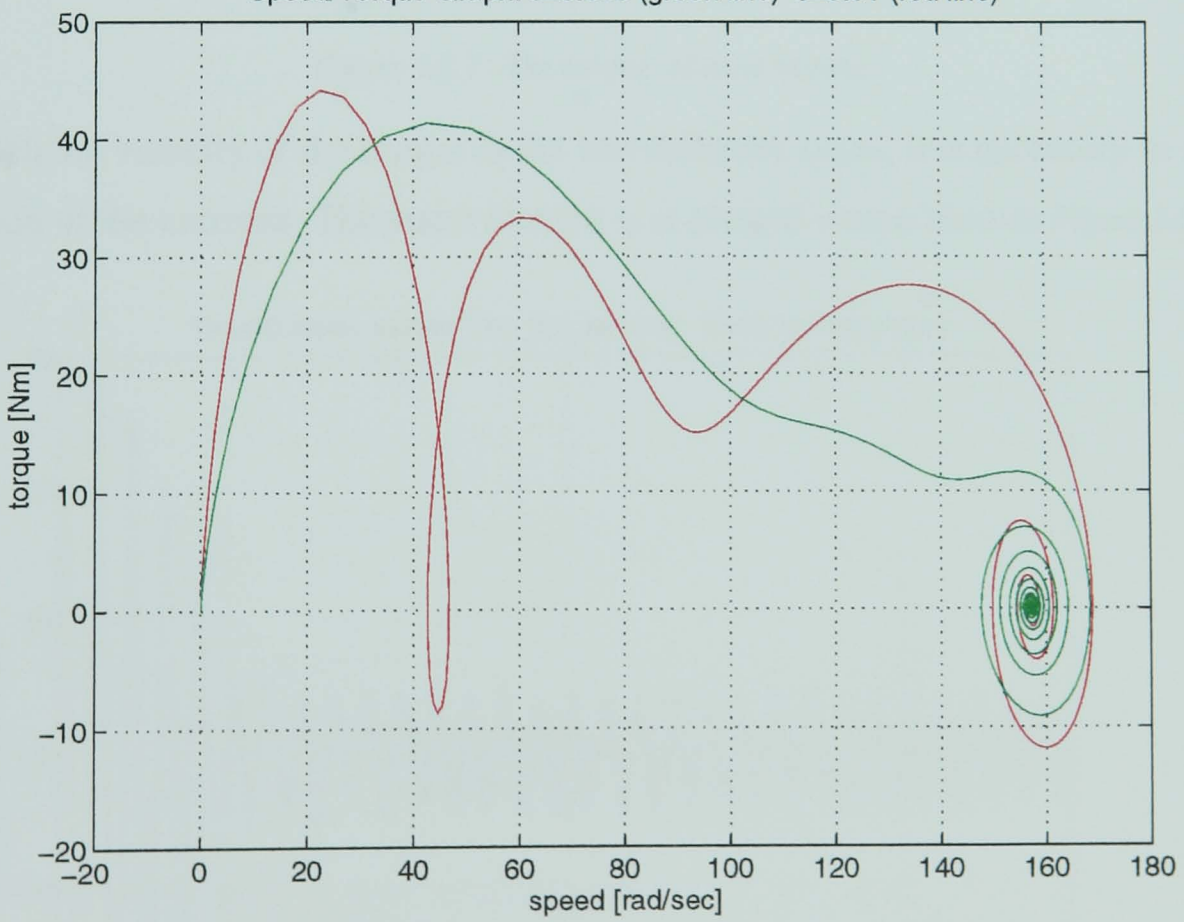

Figure 3.2.6 - Torque plotted versus speed

To further verify the *DQ*-axis model, two levels of load torque, *1* and *10 Nm* were used with the other parameters unmodified. All following figures show simulation results

plotted on the same graph for T_l=*1 Nm* and T_l=*10 Nm* . The electromagnetic torque is plotted versus time in *Figure 3.2.7*.

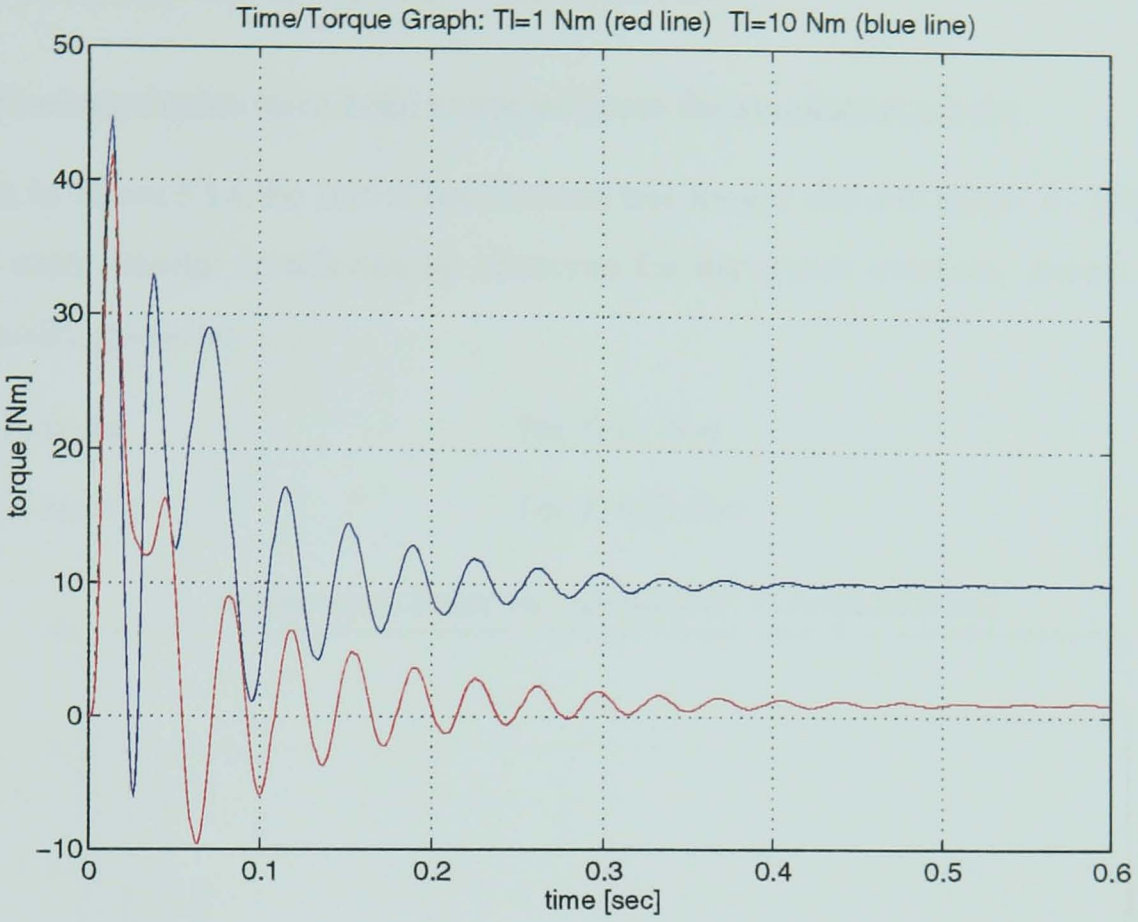

Figure 3.2.7- The torque plotted in time

To establish the veracity of the torque in the two different cases, it is necessary to study the contribution of the currents. The stator current i_D is plotted versus time in *Figure 3.2.8* .

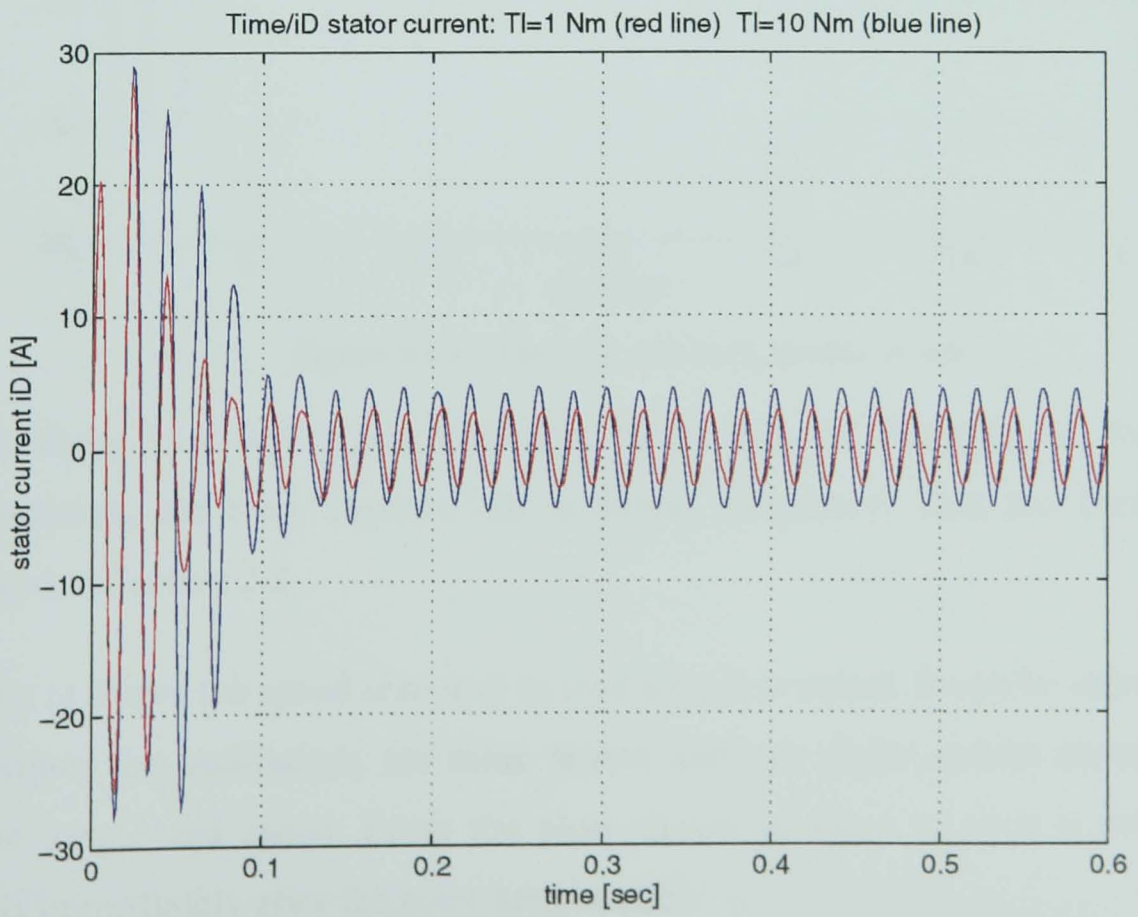

Figure 3.2.8 - The stator current i_D plotted in time

After the oscillations have decayed i_D is defined as:

$$i_{D(1)}=2.8 \sin \omega t=2.8 \cos(\omega t-\pi/2) \quad \text{for } T_l=1 \text{ Nm}$$
$$i_{D(10)}=4.45 \sin \omega t=4.45 \cos(\omega t-\pi/2) \quad \text{for } T_l=10 \text{ Nm} \qquad (3.2.4)$$

where the magnitudes have been extracted from the simulation results.

As seen in *Figure 3.2.8* the initial oscillations last longer and are larger in amplitude for the *10 Nm* case. Similar trends can be observed for the *q*-axis currents, shown in *Figure 3.2.9*, numerically given by:

$$i_{q(1)}=0.3 \sin \omega t \quad \text{for } T_l=1 \text{ Nm}$$
$$i_{q(10)}=3.3 \sin \omega t \quad \text{for } T_l=10 \text{ Nm} \qquad (3.2.5)$$

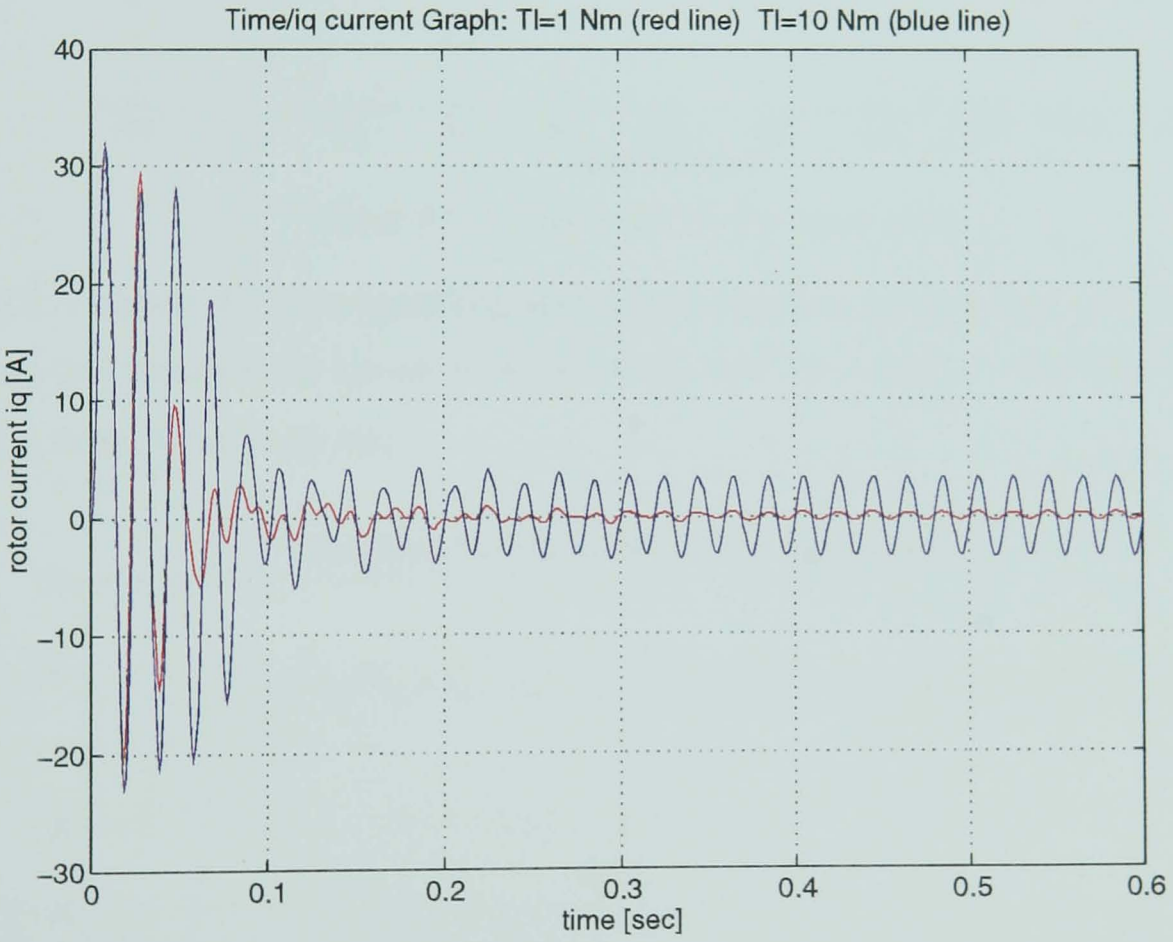

Figure 3.2.9 - The rotor current i_q plotted in time

Figures *Figure 3.2.8* and *Figure 3.2.9* also show that, as a result of the commutator transformation, the rotor currents are at supply frequency. This has been theoretically confirmed in *Section 2.2*.

Figure 3.2.11 shows the speed response as load torque is varied. It can be seen that for larger load torques the oscillations are more severe and can under certain circumstances have negative torque and speed. From the plots shown in *Figure 3.2.10* it is evident that this period is immediately after the rotor has started.

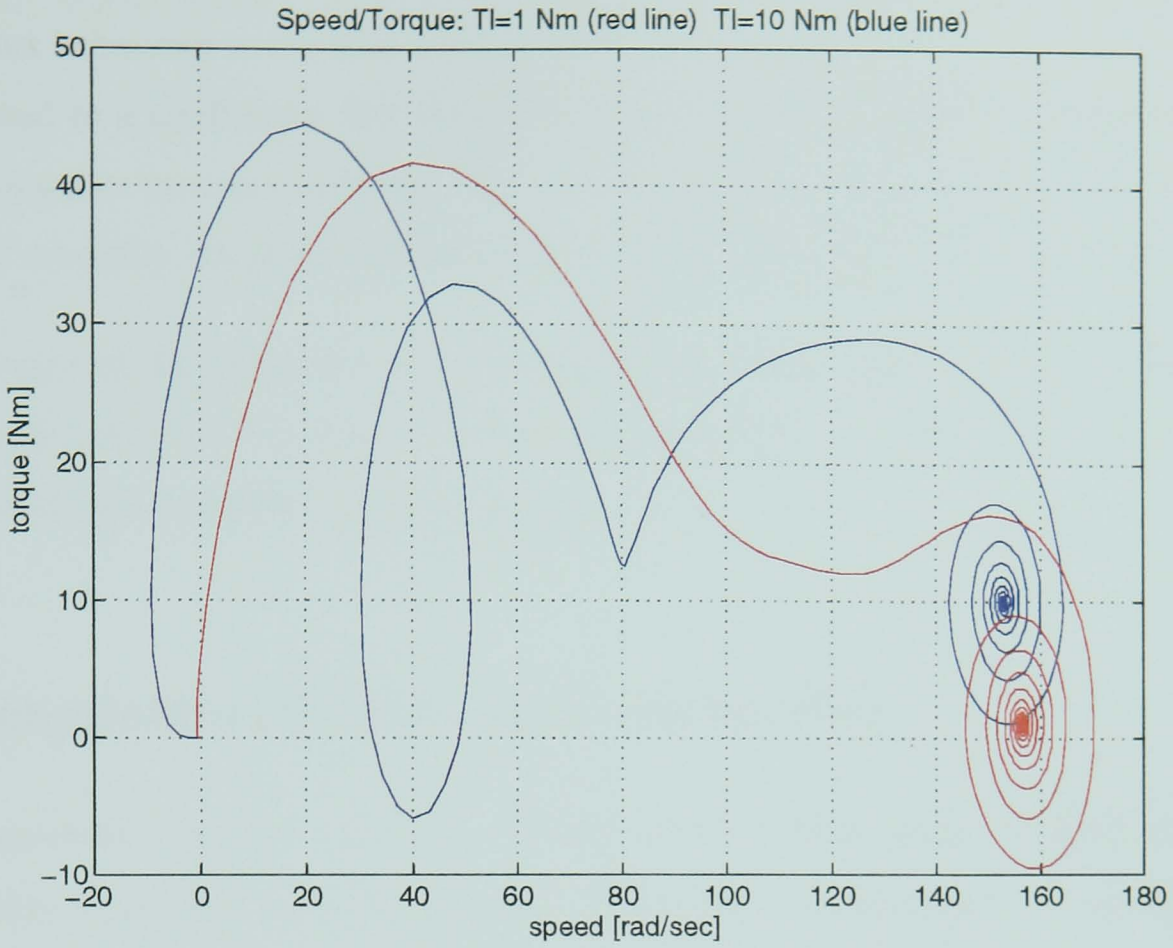

Figure 3.2.10 - The torque plotted versus speed

Figure 3.2.11 shows the corresponding speed/time response for the two different cases and confirms that steady-state speed decreases from *156.726 rad/s* to *153.0808 rad/s* as load increases from *1 Nm* to *10 Nm*.

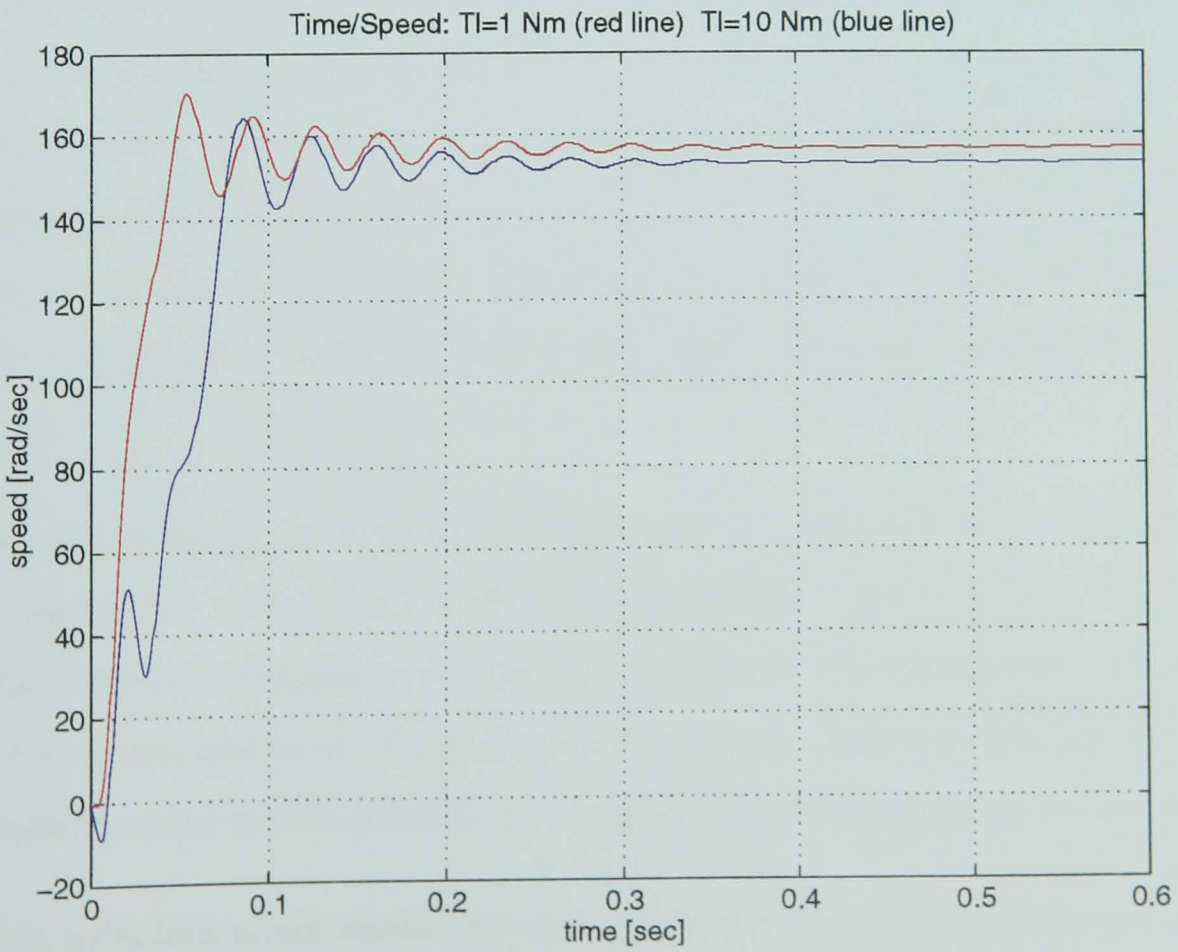

Figure 3.2.11 - The speed plotted in time

The complex behaviour in the time domain and the changes observed when the load torque is varied lead to a confidence that the induction motor model behaves satisfactory when being subjected to various conditions. Firstly, the system inertia was increased from *0.005* to *0.01* and secondly, the motor has been tested to two different levels load torque.

The next stage of the research was to modify the model to allow periodic excitations at various amplitudes and frequencies, to be imposed thereby permitting simulation of the machine under non sinusoidal input conditions.

3.3 Multi-machine DQ-axis model - introduction

In the previous sections v_D and v_Q , the stator voltages have been assumed sinusoidal quantities. In practice however the input waveform from an inverter is nonsinusoidal, the worst case being a square wave input.

The approach used in tackling the nonsinusoidal supplies is based on the superposition principle which says generally that, when a number of influences are acting upon on a system, the total influence on that particular system is given by the sum of the individual influences. Thus, if considering the PWM output harmonics as individual influences on the motor, the total effect is given by the sum of their effects. It is reminded that in *Section 2.2* some simplifying assumptions are considered which make possible the use of the superposition theorem. Saturation is not considered, however the induction motor drive model is found to work satisfactory. Furthermore the torques given by any harmonic, apart from the fundamental contribute only a very small percentage towards the value of the total torque.

In terms of modelling the superposition approach is translated into a multi-motor model which consists of individual induction motor models, each of these is excited at an individual harmonic frequency, i.e. the first is fed with the fundamental, the second with the next harmonic and so on. These harmonics when reconstituted give the inverter output. The resultant torque is then obtained by summation to estimate the overall torque.

The basic induction motor model shown in *Figure 3.2.2* can be condensed to a single block with essentially two inputs and one output. This basic block is then repeated as many times as necessary as required by the harmonic content.

This approach will be used in the next chapter which deals also with PWM modelling and PWM output decomposition.

Chapter 4

Total drive system representation

4.1 Introductory elements

A *DQ*-axis model for the induction motor has been developed in *Chapter 3*. The objective now is to model a machine drive, employing the available motor model. The approach proposed for modelling this is to use a number of induction motor models, one for each harmonic. This chapter outlines the development process undertaken for the realisation of the inverter fed induction motor model.

Consequently, a model for representing the PWM output is realised using Simulink. The block diagram and simulation results are presented next. In order to build the drive model, the PWM output has to be decomposed into a series of harmonics. This is done using Fourier Series and a sequence of functions written in the MATLAB language. To validate the results a *fft* (Fast Fourier Transform) function available in MATLAB, is used.

As all elements are available at this stage, the drive system is then realised by combining the models for the induction motor and inverter.

4.2 Modelling of the PWM output

4.2.1 Review of PWM Modelling

In order to analyse the transient and steady-state operation of electric systems, computer simulation is a very popular and economical approach. It is widely practised in industry prior to the building of the actual prototype. Through simulation, the designer can predict system behaviour over a wide range of operating conditions. It is true that customised programs are fast and efficient, however their development may take a significant amount of time and effort.

The various approaches and the computer software utilised by other researchers are reviewed and reported in the following paragraphs.

As early as 1982, Bowes and Clements [74] report on modelling systems fed with PWM waveform. A library of computer subprograms has been designed and the subroutines are written in ANSI - FORTRAN language. In 1988, Bowes and Clare report on a package called BTRAP (Bristol Transient Analysis Package). Simulation results from a motor model based on the 2-axis machine model show favourable comparison to experimental results [75].

Employing the SPICE/ORCAD software package, Maswood [76] presents models of a controlled rectifier, a current source inverter and a voltage source inverter (VSI). It is found that *PC* modelling of semiconductor switches and other nonlinear devices, can be difficult and time consuming. The so-called `*switchless*` models, based on the transfer function concept, can avoid these drawbacks.

The SPICE program is also used in [77]. SPICE has been designed mainly to study and simulate discrete device electronic circuits. It has proven to be accurate and user friendly. However does not include models of electrical machines and drives. In [77] authors propose the simulation of a switched reluctance drive using this package.

V. Rajagopalan proposes in his book [78] different methods of simulating power converters and drives. For example, a program developed by a research group at University of Quebec, Canada is called ATOSEC. With its version ATOSEC5, the book presents the simulation of a three-phase a.c. regulator-fed squirrel-cage induction motor. The simulator predicts accurately the operating characteristics of an a.c. drive system.

Another program mentioned by Martinez-Velasco and Mohan is EMTP/ATP, a general-purpose program intended for time-domain simulation of power systems [79]. It allows time-domain simulations, harmonic and frequency response analyses. The start-up of a three-phase induction machine fed from a PWM inverter is given as an illustrative case. However the experience in using this program shows that it can tackle simple cases and correcting data files can become a tedious task.

Simulink has also been employed in power electronics modelling and simulation for example by Baha. In [80] modelling of a resonant switched-mode converter is realised by employing Simulink. However as far as the author is aware, Simulink has not been previously used to model a complete drive system.

4.2.2 Inverters - PWM Technique

Variable-frequency, variable-voltage supplies using solid-state devices may be employed to control the speed of a.c. machines. There are two basic techniques for producing this type of supply: inverters and cycloconverters. The modelling of cycloconverters has been the subject of another research programme and is not discussed here.

A typical inverter has a bridge configuration with self or forced-commutation devices which are switched to produce the desired alternating voltage. The input to the bridge is the d.c. link which is obtained from a controlled or uncontrolled rectifier. In the case of a controlled rectifier, it is capable of supplying a variable d.c. voltage output.

The inverter together with the rectifier forms a converter. A schematic of a converter with a diode rectifier is shown in *Figure 4.2.1.* By using a capacitor, the input to the inverter appears as a voltage source with a very small internal impedance. The ideal case would be to have a voltage source with zero internal impedance.

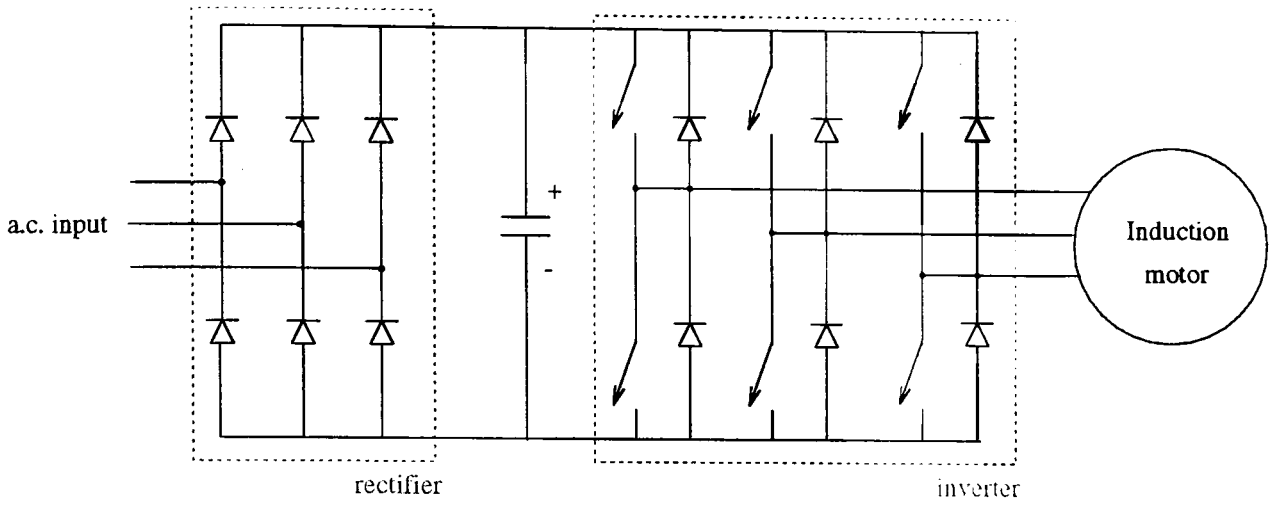

Figure 4.2.1 - Three-phase converter schematic

In *Figure 4.2.2* a general scheme for a controlled a.c. drive is presented. The three-phase a.c. input voltage has frequency f_0 and amplitude V_0 and after being rectified and inverted is applied to the a.c. motor. The converter in turn gives as an output a three-phase voltage supply characterised by *f, V*.

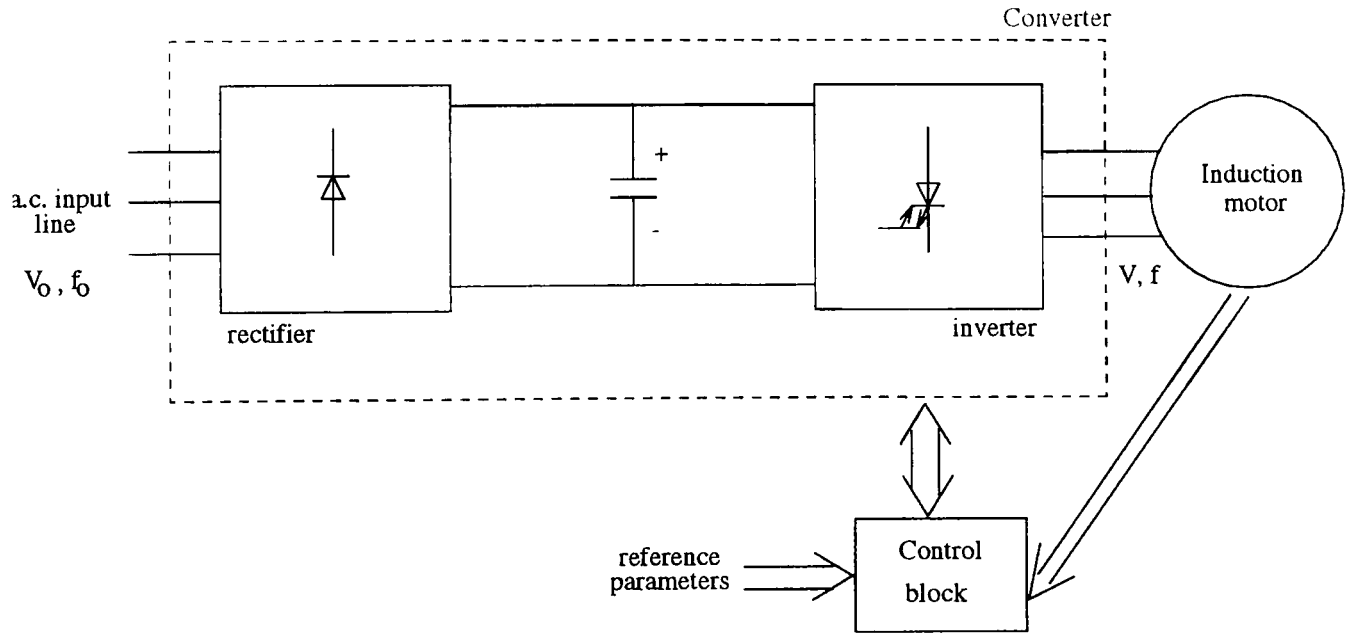

Figure 4.2.2 - An induction motor control scheme

For low to medium power the inverter is based on transistors, bipolar or IGBT (Insulated Gate Bipolar Transistor). For medium to high power range however, thyristors, normal or GTO (Gate Turn-Off thyristor) are used.

As solid-state devices have fast turn-on and turn-off times, they allow high-frequency switching and thus a train of pulses can be generated. One possibility is to have a train of pulses having variable width which can be generated using the Pulse-Width Modulation (PWM) technique.

This technique implies the production of *on/off* periods in the way that the *on*-periods are longest at the peak of the wave. The Pulse-Width Modulated output wave has a much lower low-order harmonic content that other waveforms like squarewave, quasi-squarewave or notched wave.[81].

One method of generating a PWM waveform uses an offset triangular carrier wave and a reference modulation sinewave, as shown in *Figure 4.2.3*. The inverter switching signals are generated using the intersections between the reference sinewave and the triangular waveform.

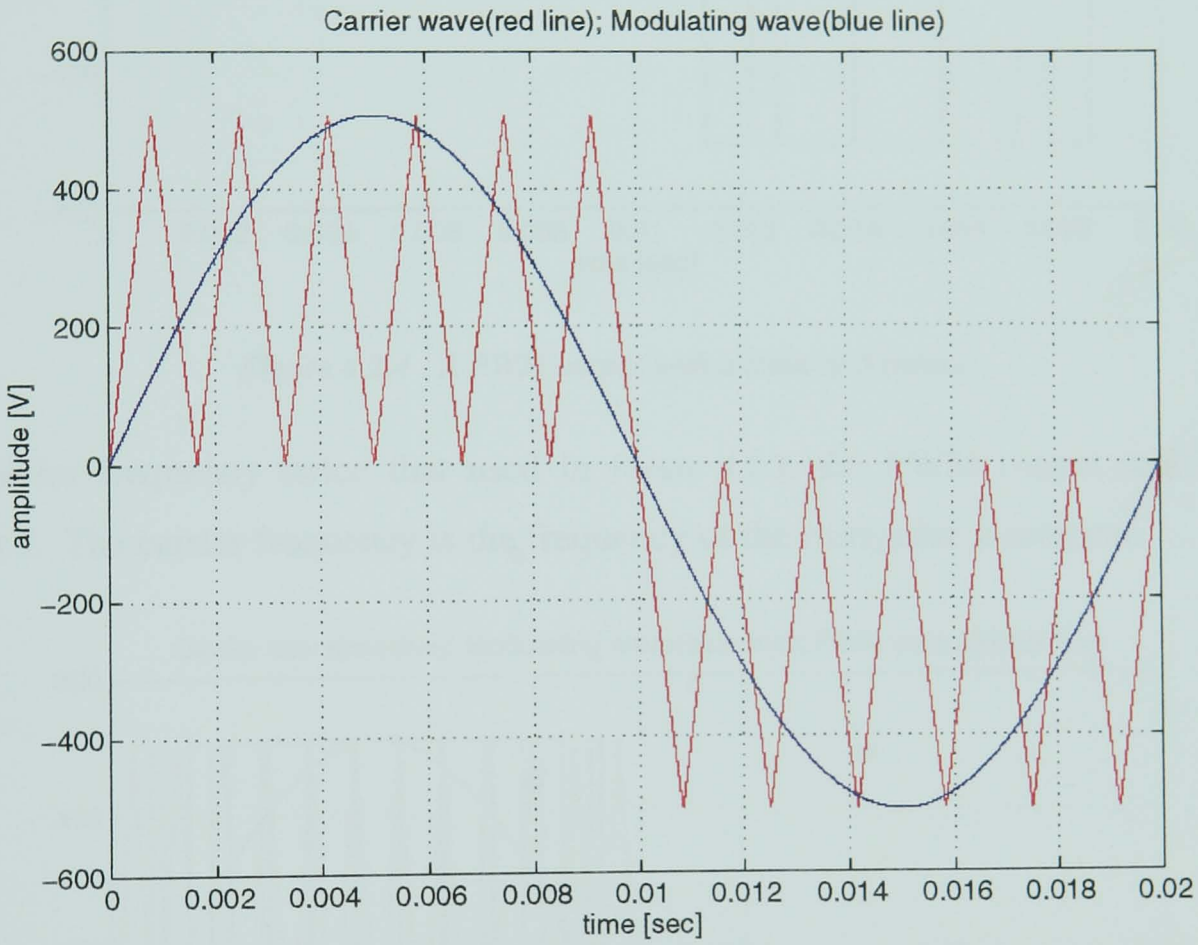

Figure 4.2.3 - The triangular carrier wave and the modulating sinewave

Applying this technique to the waveform in *Figure 4.2.3* yields the PWM output shown in *Figure 4.2.4.* which contains a large component of the fundamental frequency together with smaller components of higher frequency voltages.

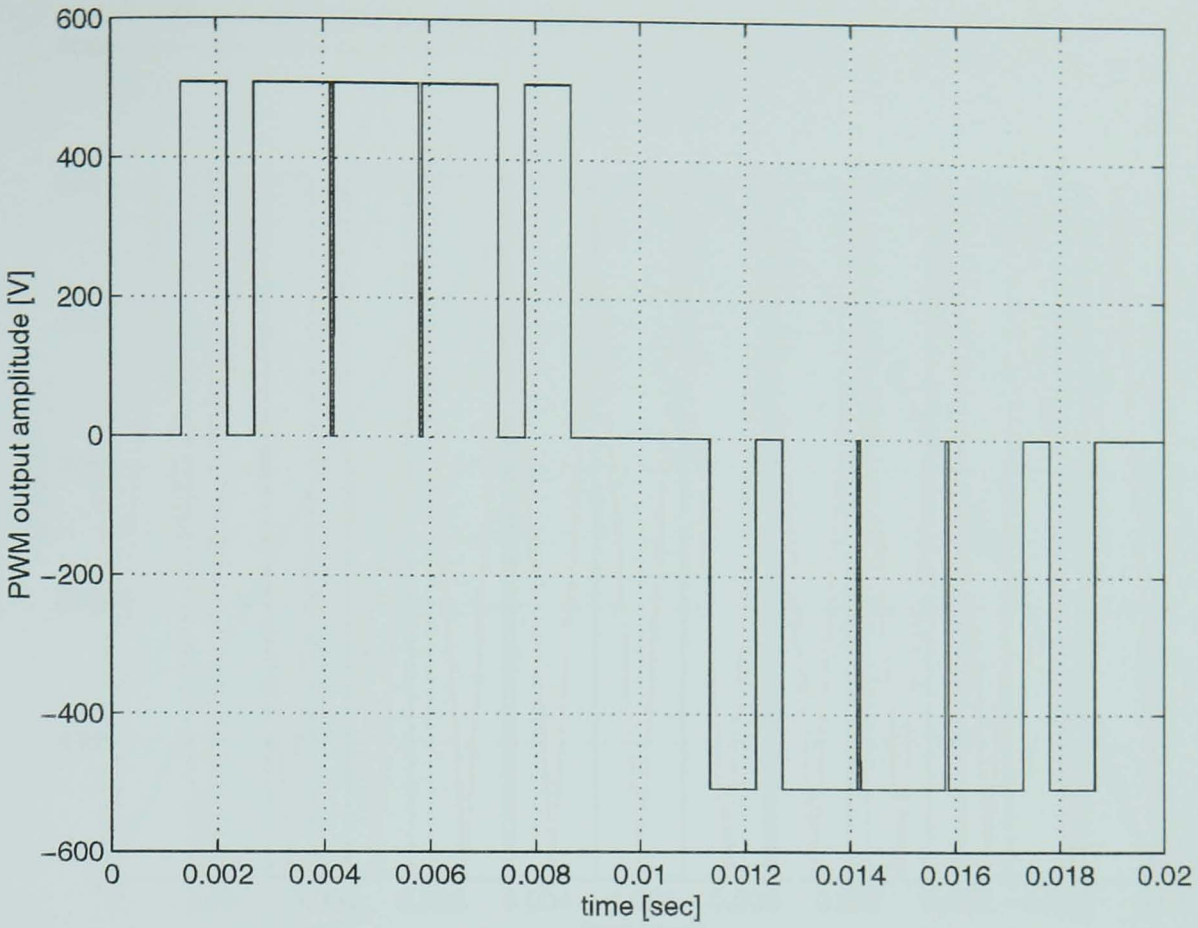

Figure 4.2.4 - A PWM output with a train of 5 pulses

For a carrier frequency twice that used in *Figure 4.2.3*, the PWM output will be as in *Figure 4.2.5*. The carrier frequency is the frequency of the triangular waveform.

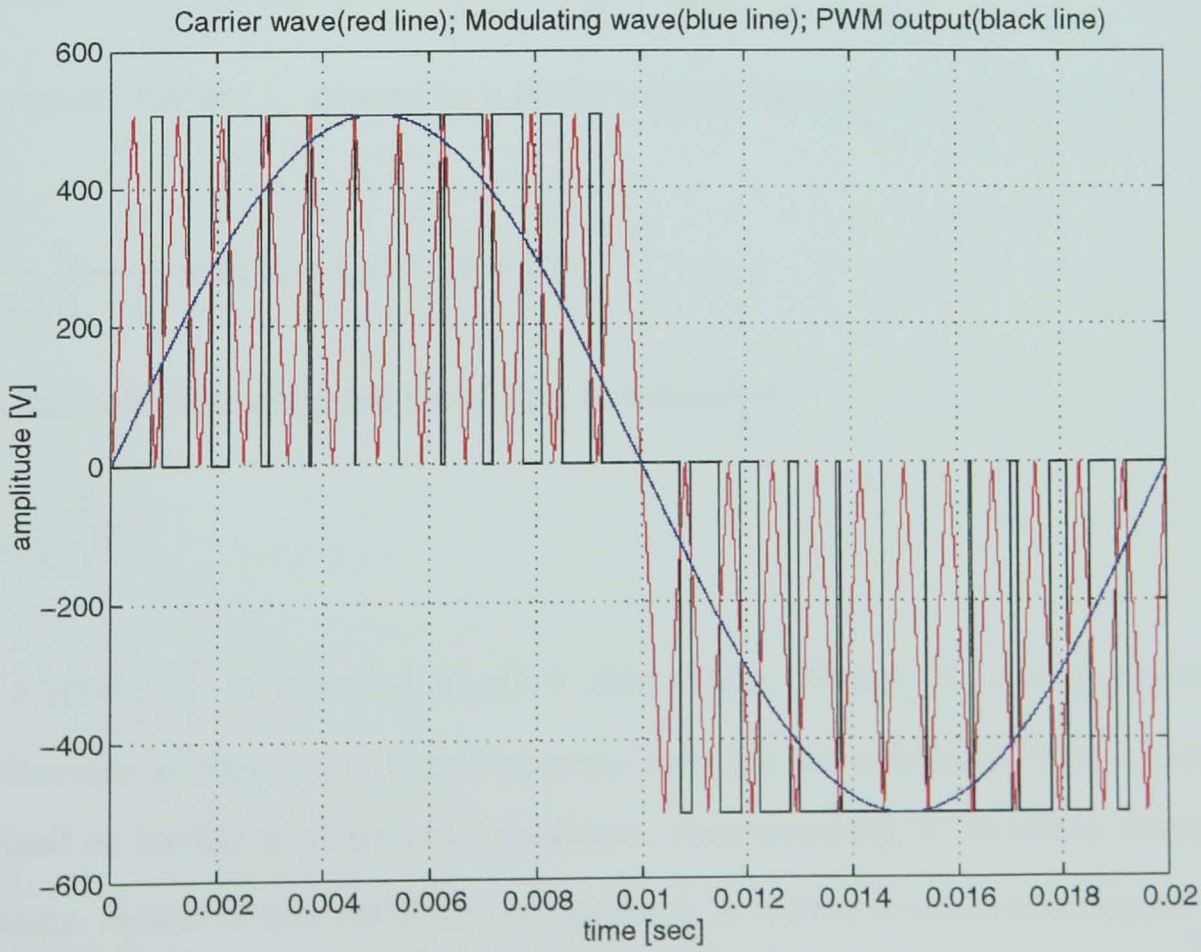

Figure 4.2.5 - A PWM output with a train of 11 pulses

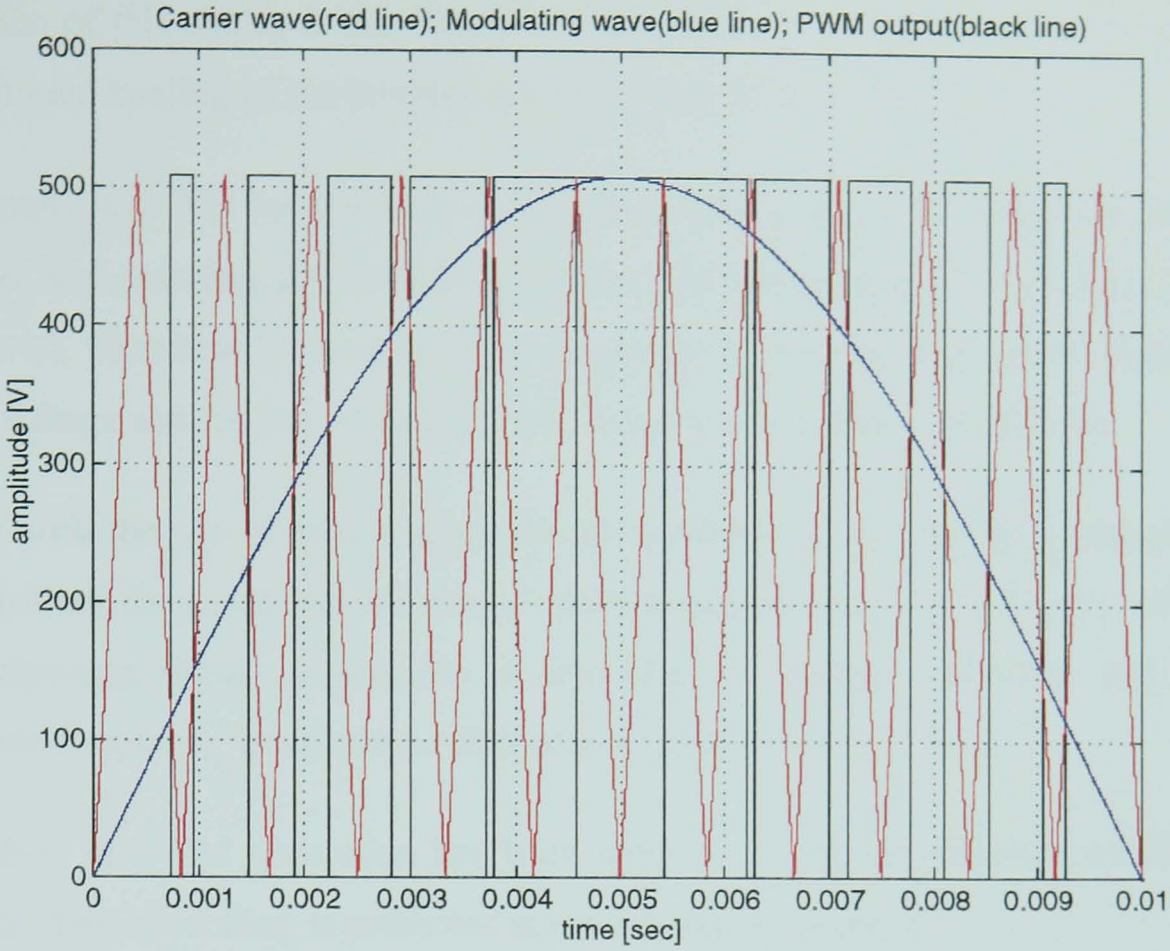

Figure 4.2.6 - Enlargement of Figure 4.2.5

In order to maintain optimum magnetic-flux conditions, below the saturation level in the induction motor, the ratio *V/f* must be kept constant. Thus any variation in the inverter frequency has to occur together with a voltage change.

If the harmonic current i_n caused by a PWM voltage harmonic v_n is given by [82]:

$$i_n \cong \frac{v_n}{2\pi n f_1 (l_s + l_r)} \; with \; (l_s + l_r) >> (R_s + R_r) \qquad (4.2.1)$$

Then a harmonically weighted loss factor is defined as:

$$\sigma = \sum \frac{v_n^2}{(n f_1)^{3/2}} \qquad (4.2.2)$$

where a term $\sqrt{n f_1}$ is included to allow skin effect. Within the motor circuit magnetic losses increase as response to high frequency voltages. An *optimum* PWM waveform could be defined as having a minimum loss factor, corresponding to minimal motor harmonic I^2R losses. However this definition of optimum is biased towards techniques which seek to minimise the motor harmonic losses, rather than minimise the inverter switching losses.

Justification of this choice is that forced cooling of the inverter heatsink is more practical than additional cooling of the inverter-driven motor [82].

The difficulty with this form of analogue control is the problem of accurately defining the two waves and ensuring that minimum *on* and *off* times required by the inverter devices are observed. Therefore it is not possible to produce three exactly balanced signals over a required voltage and frequency range when using simple analogue electronics.

However with the use of microprocessors, the problem of finding the switching instants and controlling the *on/off* periods can be tackled successfully. The difficulty which arises when employing digital modulators is achieving satisfactory resolution and operating toward zero frequency when simple chip-sets are utilised [81], [82].

This method of PWM generation has been modelled using the Simulink toolbox within MATLAB. This modelling is presented in the following section.

4.2.3 Simulink model of a PWM output

A block diagram was patched up to model the PWM output. This utilises the carrier and modulating waves to create the signals via a `*Switch*` function block.

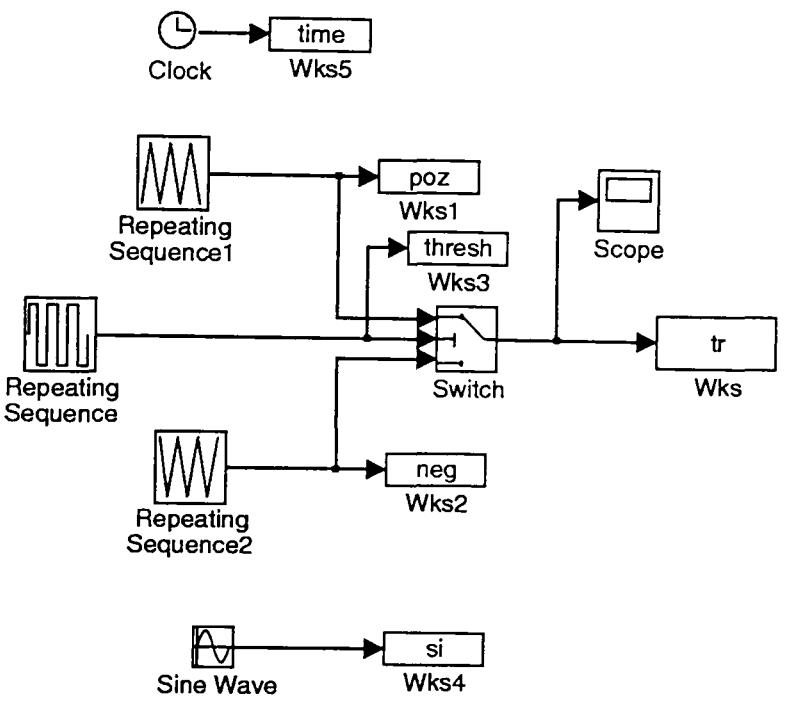

Figure 4.2.7 - The Simulink diagram for simulating the triangular wave

The block diagram of *Figure 4.2.7* simulates the triangular and the sine waves employed in obtaining the PWM output.

The PWM output is simulated by the block diagram given in *Figure 4.2.8*. There are two `*Repeating Sequence*` blocks which generate the positive and negative trains of PWM pulses respectively. To generate the `*Repeating Sequence*` functions, a number of procedures have been written in the MATLAB language. These are shown in Appendix 2. One of the key procedures developed is named *rez*, which implements the algorithm to calculate the intersection points between the triangular wave and the modulating sinusoidal wave. This algorithm is based on the root-finding routine known as the *Newton-Raphson* method. The general iterative formula for this method is given in:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad (4.2.3)$$

This method is characterised by the fact that it requires the evaluation of both the function $f(x)$ and the derivative $f'(x)$ at arbitrary points x.

In this case the function $f(x)$ is defined as the difference between a function defining the triangular wave and a sine function determining the modulating sinewave as in equation *(4.2.4)*:

$$f(x) = m \cdot x + c - vsin \cdot sin(2 \cdot \pi \cdot fq \cdot x) \qquad (4.2.4)$$

where x in this case is time, m and c are coefficients defining the triangular function, *vsin* and *fq* are the amplitude and the frequency of the modulating sinusoidal wave respectively. The procedure uses also *v*, the amplitude of the triangular wave (carrier wave) and *tri*, the number of pulses for the triangular wave per half a cycle.

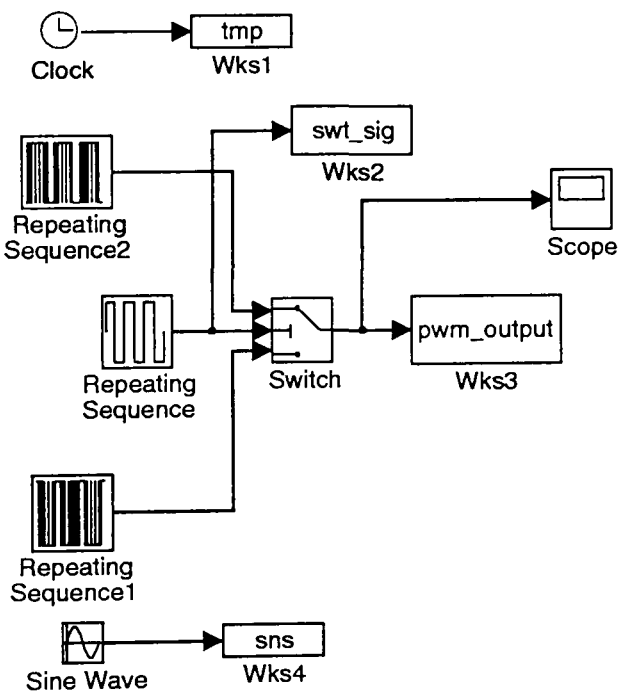

Figure 4.2.8 - The Simulink diagram for simulating the PWM output

The `*Repeating Sequence*` blocks are based on a function *mo*. This function is built on the function *rez* previously mentioned in this section. Other functions required for the `*Repeating Sequence*` blocks are: *vpoz, vneg, valori, timp*. All these functions and their purposes are listed in the Appendix 2.

In order to test the Simulink block of *Figure 4.2.8* , the following numerical case is considered : *fq*= 25 Hz, *tri*= 6, *vsin*= 254.1350 V and *v*= 508.27 V and a simulation is carried out.

Figure 4.2.9 shows the triangular and modulating waves and the resulting PWM output. Compared to the case of *Figure 4.2.4*, the amplitude of the reference sinewave is reduced to half and consequently, to observe *V/f*= constant , its frequency is also reduced.

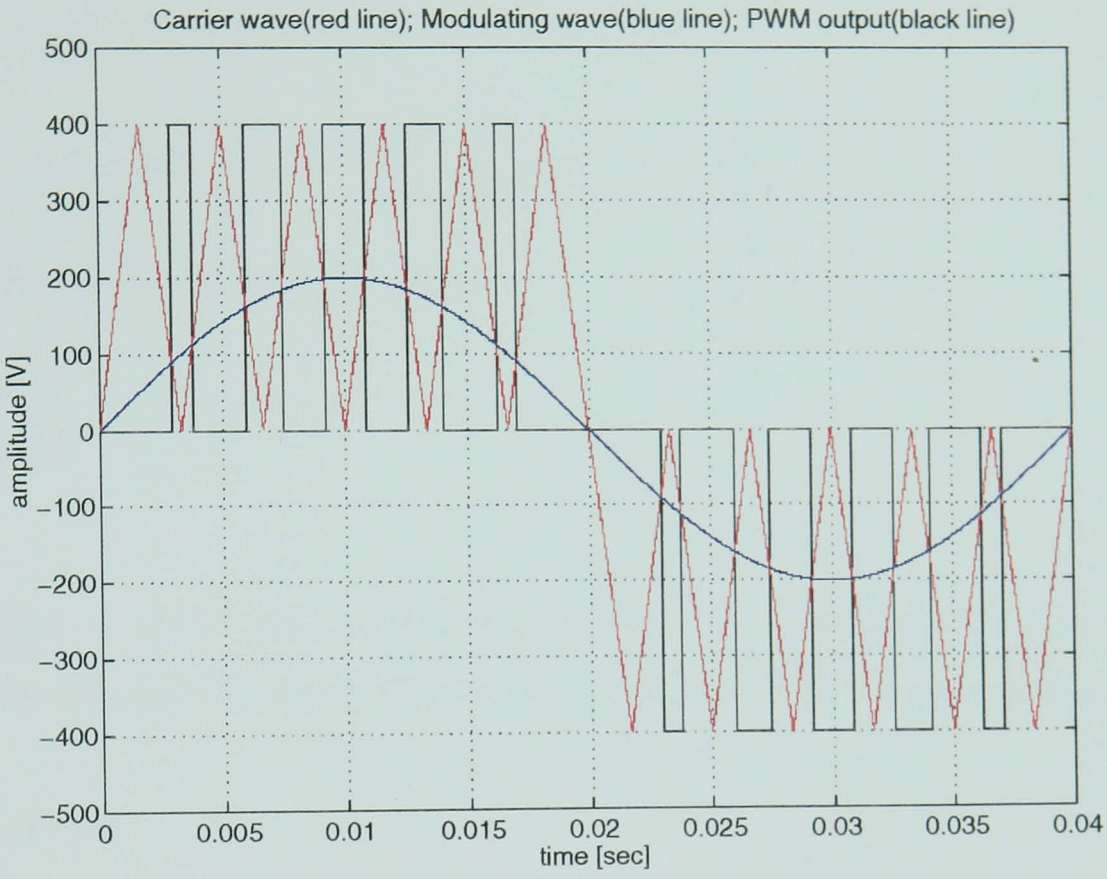

Figure 4.2.9 - The triangular wave and the modulating sinewave

The PWM output cannot be fed directly into the standard induction motor *DQ*-axis model and with respect to the improved multi-machine *DQ*-axis model there is the need for harmonics of the PWM output to be fed to the machines. The next section describes a method, based on the Fourier Series theory, to decompose the PWM output waveform into its harmonic components.

4.3 Fourier Series

Any periodical wave can be expressed as a sum of sinusoidal waves. Each sinewave has its own amplitude and its frequency is an integer multiple of the initial wave frequency. The general form of the Fourier series is given by *(4.3.1)*.

$$y = A_0 + \sum_{n=1}^{\infty} A_n \sin(n\omega t + \varphi_n) \qquad (4.3.1)$$

where $\omega = \frac{2\pi}{T}$ and T is the period. Equation *(4.3.1)* can be rewritten by expanding the sine component to give:

$$A_n \sin(n\omega t + \varphi_n) = A_n \sin\varphi_n \cos n\omega t + A_n \cos\varphi_n \sin n\omega t = a_n \cos n\omega t + b_n \sin n\omega t \qquad (4.3.2)$$

and so: $$y = \frac{1}{2} a_0 + \sum_{n=1}^{\infty} a_n \cos n\omega t + \sum_{n=1}^{\infty} b_n \sin n\omega t \qquad (4.3.3)$$

where the Fourier coefficients are given by:

$$a_n = A_n \sin \varphi_n$$
$$b_n = A_n \cos \varphi_n$$
$$a_0 = 2A_0 \qquad (4.3.4)$$

The formulas of *(4.3.1)* and *(4.3.3)* are equivalent and a periodical waveform may be expressed by either one of them. The Fourier coefficients may be calculated using [83]:

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)dx$$
$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos nxdx \qquad (4.3.5)$$
$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin nxdx$$

, where $y=f(x)$ and $x=\omega t$

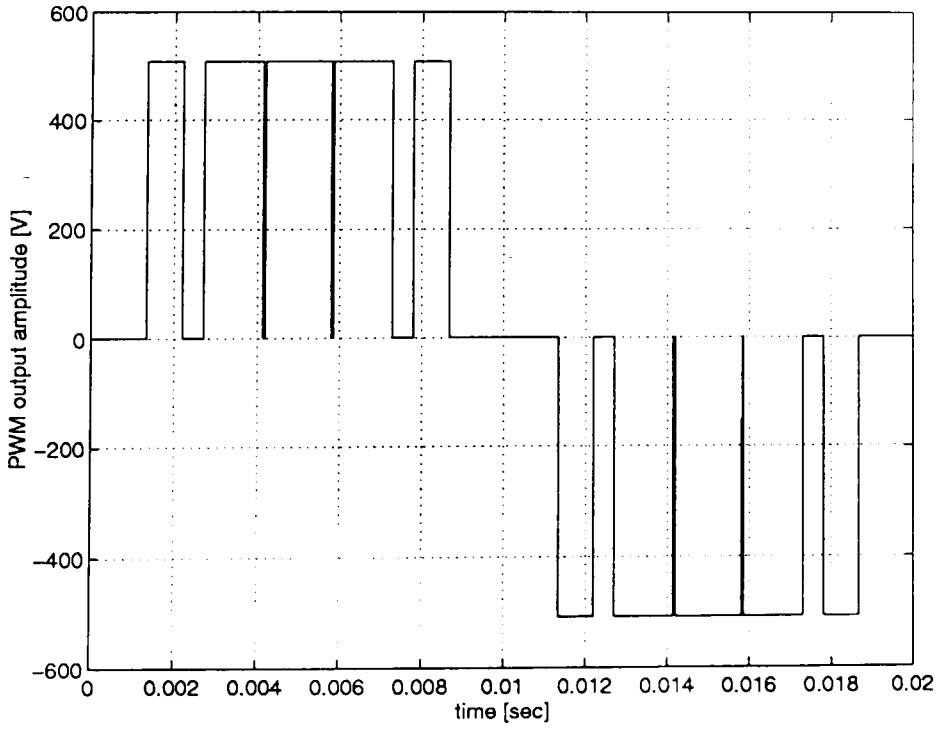

Figure 4.3.1 - The PWM output

To implement Fourier analysis for the PWM output, a typical waveform shown in *Figure 4.3.1* is considered. This waveform may be seen as consisting of five distinct sections, the first being shown in *Figure 4.3.2*. A function which describes this `*first pulse*` can be written as in:

$$f(x)=\begin{cases} 0 & for \quad [0,t_1],[t_2,\frac{1}{2fq}+t_1] \; and [\frac{1}{2fq}+t_2,\frac{1}{fq}] \\ v & for \quad [t_1,t_2] \\ -v & for \quad [\frac{1}{2fq}+t_1,\frac{1}{2fq}+t_2] \end{cases} \qquad (4.3.6)$$

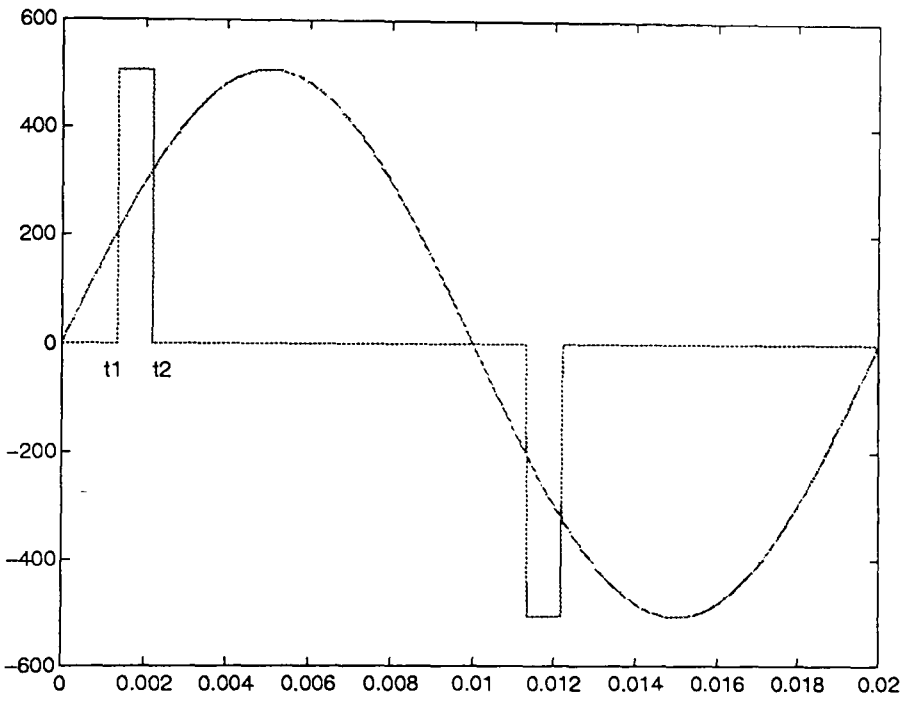

Figure 4.3.2 - A periodic function consisting of the first pulse of a PWM output

This wave is a periodic function and may be expressed as a series of sines and cosines functions and the Fourier coefficients are calculated using the formulas *(4.3.5)*, to give:

$$a_0 = 0$$

$$a_n = 0 \qquad \text{if } n \text{ is an even number}$$

$$a_n = \frac{2v}{n\pi}[\sin(n2\pi \cdot fq \cdot t_2) - \sin(n2\pi \cdot fq \cdot t_1)] \qquad \text{if } n \text{ is an odd number} \qquad (4.3.7)$$

$$b_n = 0 \qquad \text{if } n \text{ is an even number}$$

$$b_n = \frac{2v}{n\pi}[\cos(n2\pi \cdot fq \cdot t_2) - \cos(n2\pi \cdot fq \cdot t_1)] \qquad \text{if } n \text{ is an odd number} \qquad (4.3.8)$$

Since $a_0=0$ and a_n and b_n are non-zero only for odd harmonics, the function y becomes:

$$y = \sum_{n=1}^{\infty} a_n \cos n\omega t + \sum_{n=1}^{\infty} b_n \sin n\omega t \qquad \text{for } n\text{= odd number} \tag{4.3.9}$$

In explicit form, equation *(4.3.9)* is given by:

$$y = a_1 \cos \omega t + a_3 \cos 3\omega t + a_5 \cos 5\omega t + \ldots \\ + b_1 \sin \omega t + b_3 \sin 3\omega t + a_5 \sin 5\omega t + \ldots \tag{4.3.10}$$

The numerical example chosen is : v=508.27 (the amplitude of the carrier wave), fq=50 (the frequency of the modulating wave), $vsin$=508.27 (the amplitude of the modulating wave), tri=6 (the ratio between the carrier and the modulating wave frequencies). These values produce the PWM output shown in *Figure 4.3.1.* The function *rez* [*Section 4.2.2*] invoked at the prompt in MATLAB command window gives the following results:

```
>> rez(50,508.27,508.27,6)

ans =

         0
    0.0013
    0.0022
    0.0027
    0.0041
    0.0042
    0.0058
    0.0059
    0.0073
    0.0078
    0.0087
    0.0100

>>
```

Figure 4.3.3- results of invoking **rez** *in MATLAB command window*

These results give the switching instants for the PWM output waveform of *Figure 4.3.1.* Therefore t_1 and t_2 [*Figure 4.3.2*] can be evaluated. It can bee seen from the *Figure 4.3.3* that: t_1=0.0013 and t_2=0.0022. Substituting in the relevant equations allows the Fourier coefficients, a_1 and b_1 to be calculated as:

$$a_1 = \frac{2 \cdot 508.27}{\pi}[\sin(2\pi \cdot 50 \cdot 0.0022) - \sin(2\pi \cdot 50 \cdot 0.0013)] = 77.7473 \tag{4.3.11}$$

$$b_1 = \frac{2 \cdot 508.27}{\pi}[\cos(2\pi \cdot 50 \cdot 0.0022) - \cos(2\pi \cdot 50 \cdot 0.0013)] = -47.6436 \tag{4.3.12}$$

Higher order coefficients may be also calculated and values for the first five harmonics are given in *Table 4.3.1.*

	the Fourier a coefficient	*the Fourier b coefficient*
1st harmonic	$a_1 = 77.7473$	$b_1 = -47.6436$
3rd harmonic	$a_3 = -6.9648$	$b_3 = -88.4968$
5th harmonic	$a_5 = -77.6595$	$b_5 = -32.1676$
7th harmonic	$a_7 = -58.7568$	$b_7 = 50.1831$
9th harmonic	$a_9 = 16.0439$	$b_9 = 66.8278$

Table 4.3.1- Fourier coefficients for the waveform of Figure 4.3.2 ('first pulse')

In equation *(4.3.9)* the harmonic number n tends to infinity. In practical situations the maximum value for n is chosen depending on the desired level of approximation to be achieved.

Obviously, the same procedure can be applied to the second and subsequent pulses of the PWM output. For example in *Figure 4.3.4*, t_3 and t_4 are used instead of t_1 and t_2.

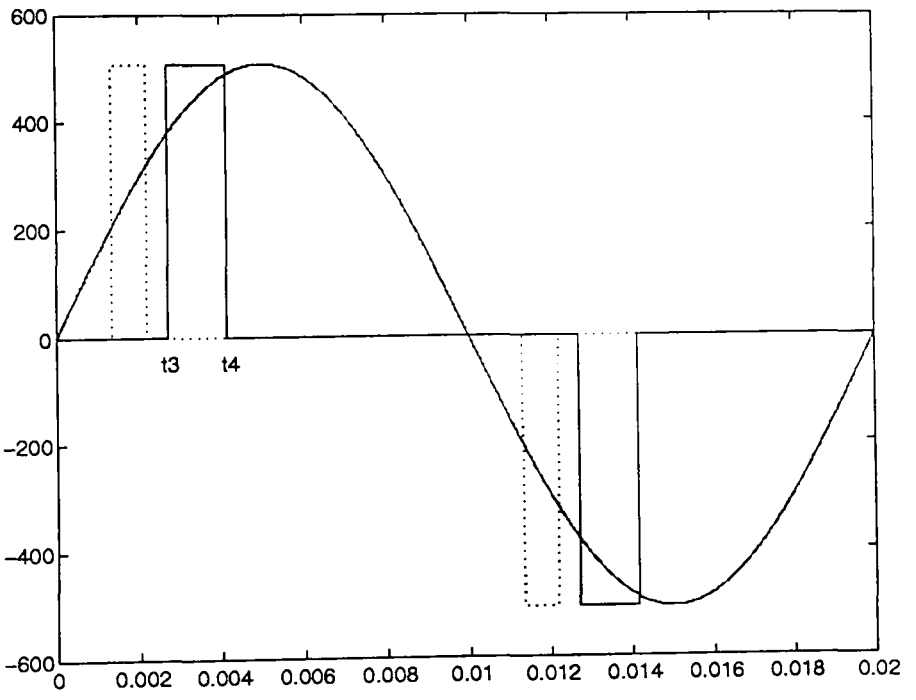

Figure 4.3.4 - The `second pulse` in a PWM output

The expression for the second pulse follows:

$$y_{(2)} = a_{1_{(2)}} \cos \omega t + a_{3_{(2)}} \cos 3\omega t + a_{5_{(2)}} \cos 5\omega t + \ldots + b_{1_{(2)}} \sin \omega t + b_{3_{(2)}} \sin 3\omega t + a_{5_{(2)}} \sin 5\omega t + \ldots \qquad (4.3.13)$$

where the subscript *(2)* has been used to represent the second pulse. The corresponding Fourier coefficients are evaluated as:

$$a_{1_{(2)}} = (2 \cdot 508.27)/n \cdot [\sin(2 \cdot \pi \cdot 50 \cdot 0.0041) - \sin(2 \cdot \pi \cdot 50 \cdot 0.0027)] = 68.0098 \qquad (4.3.14)$$

	the Fourier a coefficient	*the Fourier b coefficient*
1st harmonic	$a_{1(2)}$= 68.0098	$b_{1(2)}$= -123.7093
3rd harmonic	$a_{3(2)}$= -131.9533	$b_{3(2)}$= 8.3018
5th harmonic	$a_{5(2)}$= 67.7851	$b_{5(2)}$= 93.2982
7th harmonic	$a_{7(2)}$= 34.0163	$b_{7(2)}$= -85.9153
9th harmonic	$a_{9(2)}$= -64.8227	$b_{9(2)}$= 12.3656

Table 4.3.2- Fourier coefficients for the 2nd pulse

This procedure is extended to the remaining pulses so that the PWM waveform is now given by:

$$y = y_{(1)} + y_{(2)} + y_{(3)} + y_{(4)} + y_{(5)} \tag{4.3.15}$$

and in an expanded form:

$$\begin{aligned} y = {} & a_{1_{(1)}} \cos \omega t + a_{3_{(1)}} \cos 3\omega t + a_{5_{(1)}} \cos 5\omega t + \ldots \\ & + b_{1_{(1)}} \sin \omega t + b_{3_{(1)}} \sin 3\omega t + a_{5_{(1)}} \sin 5\omega t + \ldots \\ & + a_{1_{(2)}} \cos \omega t + a_{3_{(2)}} \cos 3\omega t + a_{5_{(2)}} \cos 5\omega t + \ldots \\ & + b_{1_{(2)}} \sin \omega t + b_{3_{(2)}} \sin 3\omega t + a_{5_{(2)}} \sin 5\omega t + \ldots \\ & + a_{1_{(3)}} \cos \omega t + a_{3_{(3)}} \cos 3\omega t + a_{5_{(3)}} \cos 5\omega t + \ldots \\ & + b_{1_{(3)}} \sin \omega t + b_{3_{(3)}} \sin 3\omega t + a_{5_{(3)}} \sin 5\omega t + \ldots \\ & + a_{1_{(4)}} \cos \omega t + a_{3_{(4)}} \cos 3\omega t + a_{5_{(4)}} \cos 5\omega t + \ldots \\ & + b_{1_{(4)}} \sin \omega t + b_{3_{(4)}} \sin 3\omega t + a_{5_{(4)}} \sin 5\omega t + \ldots \\ & + a_{1_{(5)}} \cos \omega t + a_{3_{(5)}} \cos 3\omega t + a_{5_{(5)}} \cos 5\omega t + \ldots \\ & + b_{1_{(5)}} \sin \omega t + b_{3_{(5)}} \sin 3\omega t + a_{5_{(5)}} \sin 5\omega t + \ldots \end{aligned} \tag{4.3.16}$$

Rearranging equation *(4.3.16)* gives:

$$\begin{aligned} y = {} & \cos\omega t(a_{1_{(1)}} + a_{1_{(2)}} + a_{1_{(3)}} + a_{1_{(4)}} + a_{1_{(5)}}) + \cos 3\omega t(a_{3_{(1)}} + a_{3_{(2)}} + a_{3_{(3)}} + a_{3_{(4)}} + a_{3_{(5)}}) + \ldots \\ & + \sin\omega t(b_{1_{(1)}} + b_{1_{(2)}} + b_{1_{(3)}} + b_{1_{(4)}} + b_{1_{(5)}}) + \sin 3\omega t(b_{3_{(1)}} + b_{3_{(2)}} + b_{3_{(3)}} + b_{3_{(4)}} + b_{3_{(5)}}) + \ldots \end{aligned} \tag{4.3.17}$$

where:

$$a_1 = a_{1_{(1)}} + a_{1_{(2)}} + a_{1_{(3)}} + a_{1_{(4)}} + a_{1_{(5)}} \tag{4.3.18}$$

$$b_1 = b_{1_{(1)}} + b_{1_{(2)}} + b_{1_{(3)}} + b_{1_{(4)}} + b_{1_{(5)}} \quad \text{etc.} \tag{4.3.19}$$

and similarly the other coefficients can be evaluated. Subscripts *(1)*, *(2)*, *(3)*, *(4)* and *(5)* refer to each of the five `*one-pulse*` functions respectively.

The PWM waveform has been expressed in the form of *(4.3.3)*. Knowing a_n, b_n and that the PWM waveform has no d.c. component, i.e. A_0=0, the Fourier series may be represented by *(4.3.1)*:

$$y = \sum_{n=1}^{\infty} A_n \sin(n\omega t + \varphi_n) \qquad (4.3.20)$$

where the amplitude and the phase angle for each harmonic can be calculated using:

$$tg\varphi_n = \frac{a_n}{b_n} \qquad (4.3.21)$$

$$A_n = \frac{a_n}{\sin\varphi_n} = \frac{a_n}{\frac{tg\varphi_n}{\sqrt{1+tg^2\varphi_n}}} \qquad (4.3.22)$$

For the example considered the first 5 coefficients are given in *Table 4.3.3*.

	the Fourier A coefficient	***the phase angle***
1st harmonic	A_1= 508.2691	φ_1= $-0.23*10^{-13}=0$
3rd harmonic	A_3= -0.0404	φ_3= $-0.55*10^{-9}=0$
5th harmonic	A_5= -1.1067	φ_5= $-0.98*10^{-12}=0$
7th harmonic	A_7= -16.8716	φ_7= $0.19*10^{-11}=0$
9th harmonic	A_9= -107.9006	φ_9= $-0.01*10^{-11}=0$

Table 4.3.3 - Fourier coefficients

Therefore *y* is given by A_n and φ_n which were calculated based on a_n and b_n [*4.3.21-22*]. To calculate automatically A_n and φ_n , some functions have been written in the MATLAB language. These are *An* , to calculate the amplitudes and *faza* , to calculate the phase angles of the harmonic components. Their parameters are: *v* , *vsin* , *tri* . *fq* , *nr*. The first four parameters have been previously defined. The *nr* parameter is the number of frequency components selected to be considered. For example, when *An(50,508.27,508.27,6,7)* the following results are obtained:

```
>> An(50,508.27,508.27,6,7)

ans =

  508.2691
   -0.0404
   -1.1067
  -16.8716
 -107.9006
  -92.1310
   91.5850

>>
```

Figure 4.3.5- results of An invoking in MATLAB command window

Computation of the phase angle is obtained by calling:

```
>> faza(50,508.27,508.27,6,7)

ans =

   1.0e-09 *

   -0.0000
   -0.5588
   -0.0010
    0.0019
   -0.0001
   -0.0004
   -0.0002

>>
```

*Figure 4.3.6- results of **faza** procedure invoked in MATLAB window*

which give:

	the Fourier A_n coefficient	*the phase angle φ_n*
1st harmonic	A_1= 508.2691	φ_1= -0.00001*10^{-9} rad
3rd harmonic	A_3= -0.0404	φ_3= -0.5588*10^{-9} rad
5th harmonic	A_5= -1.1067	φ_5= -0.0010*10^{-9} rad
7th harmonic	A_7= -16.8716	φ_7= 0.0019*10^{-9} rad
9th harmonic	A_9= -107.9006	φ_9= -0.0001*10^{-9} rad
11th harmonic	A_{11}= -92.1310	φ_{11}= -0.0004*10^{-9} rad
13th harmonic	A_{13}= 91.5850	φ_{13}= -0.0002*10^{-9} rad

Table 4.3.4 - Fourier coefficients

The objective of developing these functions (*An* and *faza*) is to allow a drive model to be automatically built by a `*m file*` written in MATLAB language. This drive model will be presented in the following *Section 4.5.*

In order to verify the validity of PWM output decomposition in harmonics and to inspect the reconstituted PWM wave, the first *50* harmonics, i.e. the 1^{st}, the 3^{rd} , the 5^{th} ...and the 99^{th} are summed together. The amplitude and angle information are given by the functions *An* and *faza* .

```
>> An(50,508.27,508.27,6,7)

ans =

  508.2691
   -0.0404
   -1.1067
  -16.8716
 -107.9006
  -92.1310
   91.5850

>>
```

*Figure 4.3.5- results of **An** invoking in MATLAB command window*

Computation of the phase angle is obtained by calling:

```
>> faza(50,508.27,508.27,6,7)

ans =

   1.0e-09 *

   -0.0000
   -0.5588
   -0.0010
    0.0019
   -0.0001
   -0.0004
   -0.0002

>>
```

*Figure 4.3.6- results of **faza** procedure invoked in MATLAB window*

which give:

	the Fourier A_n coefficient	***the phase angle φ_n***
1st harmonic	$A_1 = 508.2691$	$\varphi_1 = -0.00001*10^{-9}$ rad
3rd harmonic	$A_3 = -0.0404$	$\varphi_3 = -0.5588*10^{-9}$ rad
5th harmonic	$A_5 = -1.1067$	$\varphi_5 = -0.0010*10^{-9}$ rad
7th harmonic	$A_7 = -16.8716$	$\varphi_7 = 0.0019*10^{-9}$ rad
9th harmonic	$A_9 = -107.9006$	$\varphi_9 = -0.0001*10^{-9}$ rad
11th harmonic	$A_{11} = -92.1310$	$\varphi_{11} = -0.0004*10^{-9}$ rad
13th harmonic	$A_{13} = 91.5850$	$\varphi_{13} = -0.0002*10^{-9}$ rad

Table 4.3.4 - Fourier coefficients

The objective of developing these functions (*An* and *faza*) is to allow a drive model to be automatically built by a `*m file*` written in MATLAB language. This drive model will be presented in the following *Section 4.5.*

In order to verify the validity of PWM output decomposition in harmonics and to inspect the reconstituted PWM wave, the first *50* harmonics, i.e. the 1^{st}, the 3^{rd} , the 5^{th} ...and the 99^{th} are summed together. The amplitude and angle information are given by the functions *An* and *faza* .

The synthesised PWM output is shown in *Figure 4.3.7.* It can be seen that the original PWM output waveform has not been approximated accurately.

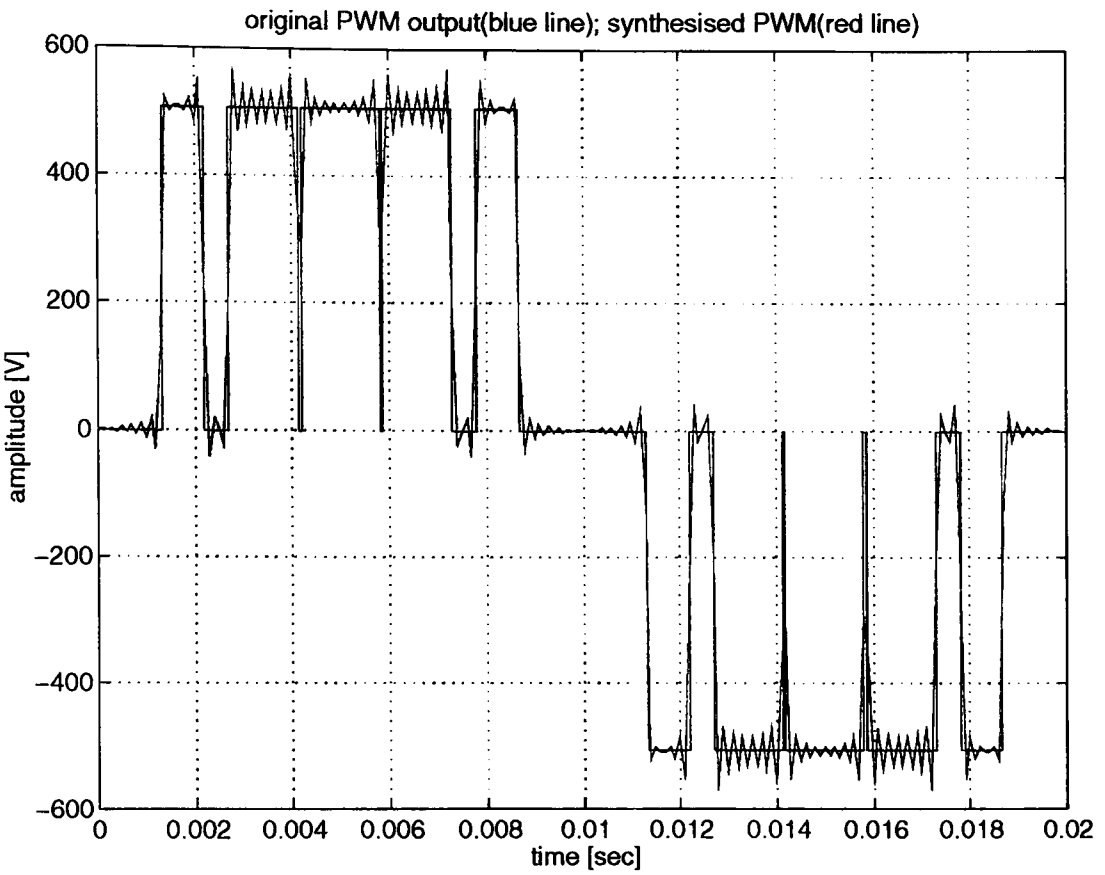

Figure 4.3.7 - The regained PWM output

This behaviour is due to two effects, firstly, the total number of harmonics is not adequate since this should tend towards infinity. Secondly, the original waveform has discontinuities, so that when the Fourier terms are summed, every original point is regained except at the discontinuities, where an overshoot will occur. This behaviour encountered at discontinuity points is known as the Gibb's phenomenon [83].

In the following section results given by invoking *An* and *faza* are compared against results obtained using the MATLAB built-in function *fft*.

4.4 FFT (Fast Fourier Transform)

An alternative approach to decomposing the PWM output waveform is to use the built-in MATLAB function *fft* (*Fast Fourier Transform*). This is based on an algorithm with the same name. It is actually the discrete Fourier transform of a vector. Therefore the periodical wave to be decomposed has to be windowed and sampled.

A feature of the *fft* approach is that the sampling frequency is important for the *fft* results.

A proper sampling must obey the *Nyquist* sampling theorem which says that the sampling rate should be at least twice the frequency of the highest frequency component in the waveform being sampled [83]. The need to simulate first the waveform does not allow an automatically implementation of a motor drive model.

However, here the results given by the *fft* are used to certify those obtained by *An* and *faza* functions.

The results obtained by applying *fft* to a function are in rectangular form and consequently as the results are needed in polar form, they require to be transformed using the relationships:

$$magnitude = \sqrt{Re^2 + Im^2} \tag{4.4.1}$$

$$phase = arctan(\frac{Im}{Re}) \tag{4.4.2}$$

where *Im* and *Re* are respectively the imaginary and real parts of the values obtained. All the calculations are incorporated in a small program called *progfft* which decomposes the function and then plots the results. The *fft* is applied to the function representing the PWM output of *Figure 4.3.1* .The values A_n and φ_n obtained are given in *Figure 4.4.1.*, where the A_n values are on the left side.

```
  508.2691         0
    0.0166    3.1416
    0.9771    3.1416
   16.8522    3.1416
  108.0567    3.1416
   92.1358    3.1416
   91.6319         0

>>
```

Figure 4.4.1 - A_n and φ_n *using the MATLAB fft function*

therefore *y* can be evaluated as:

$$y = 508.2691\sin(\omega t + 0) + 0.0166\sin(3\omega t + 3.1416) + 0.9771\sin(5\omega t + 3.1416) + \ldots \tag{4.4.3}$$

where $\omega = 2\pi f$, $f = 50\ Hz$.

Calculating with *(4.3.20)* or with *(4.4.3)* show similar results. This proves that the functions written in MATLAB language *An* and *faza* are correct, giving approximately the same results as the *fft* function.

As mentioned earlier, a higher inverter switching frequency gives a PWM output with a better harmonic content. Some simulations are performed using the Simulink block diagram from *Figure 4.2.8* and PWM outputs are obtained for two different switching frequencies. Using the *fft* function and the MATLAB written program *progfft* , the PWM waveforms are translated from the time domain to the frequency domain.

For the *300 Hz* switching frequency PWM waveform, 4 harmonics can be noticed to have amplitudes about one fifth of the fundamental amplitude. These are the 9^{th} harmonic, the 11^{th}, the 13^{th} and the 15^{th}. The PWM output can be seen in *Figure 4.4.2.*

For the *600 Hz* PWM a translation of those harmonics to the right on the frequency-axis is encountered. Thus the harmonics with high amplitudes are now : the 21^{st} harmonic, the 23^{rd} , the 25^{th} and the 27^{th}. The resulting graphs show that Pulse-Width Modulation can move unwanted frequency components to a higher frequency region. The *600 Hz* PWM waveform is shown in *Figure 4.4.3.*

A further quality indicator, the total harmonic distortion factor, can be defined as being the ratio of the rms value of all harmonic components together to the rms amplitude of the fundamental [81]:

$$THD = \frac{\sqrt{V_{3(rms)}^2 + V_{5(rms)}^2 + V_{7(rms)}^2 + \ldots + V_{n(rms)}^2}}{V_{1(rms)}} \qquad (4.4.4)$$

Estimation of THD using only the harmonics visible in the figures, the first 25, give:

THD (*300 Hz*) = 44.49 %

THD (*600 Hz*) = 42.89 %

Also THD was calculated for *900 Hz* and *1200 Hz*, and found to be :

THD (*900 Hz*) = 39.75 %

THD (*1200 Hz*) = 33.44 %

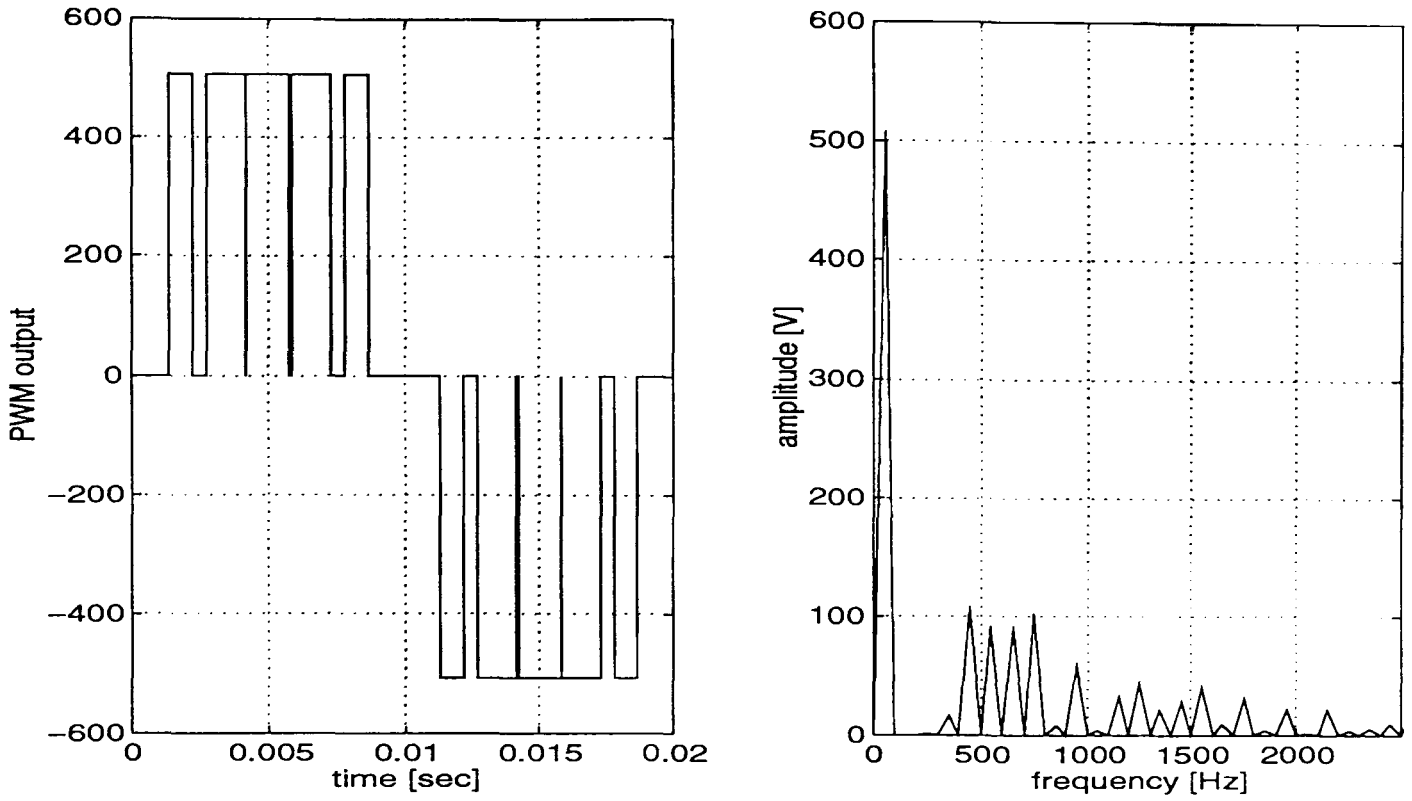

Figure 4.4.2 - A 5 pulse PWM output waveform in time and frequency domain

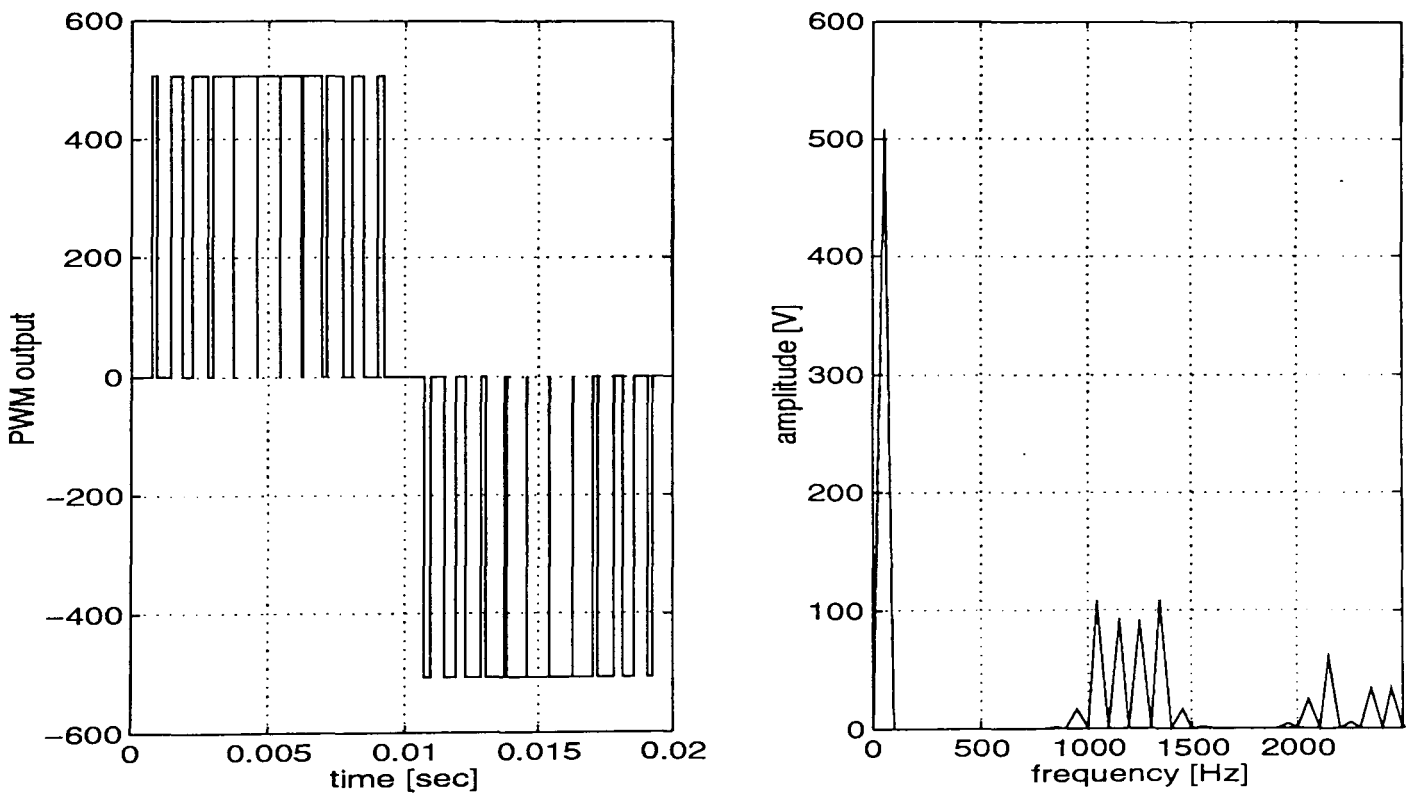

Figure 4.4.3. - An 11 pulse PWM output waveform in time and frequency domain

Therefore the output waveform of a PWM inverter is seen to be generally improved by using a high ratio between the carrier frequency and the output fundamental frequency. With the induction motor model available from *Chapter 3* and the PWM output harmonics determined, the next step in model developments is to combine them together to simulate the total drive system.

4.5 The total drive system model and simulation results

It has been shown that the PWM output waveform can be decomposed in to a series of harmonics. These harmonic components can be applied to the inverter fed induction machine model described in *Chapter 3*. The block diagram in *Figure 2.5.2* for a `*single*` motor model is simplified to that shown in *Figure 4.5.1*, by compressing the block models for the four voltage equations of *(2.5.9)* system. These are now shown as *EQ1*, *EQ2*, *EQ3* and *EQ4*.

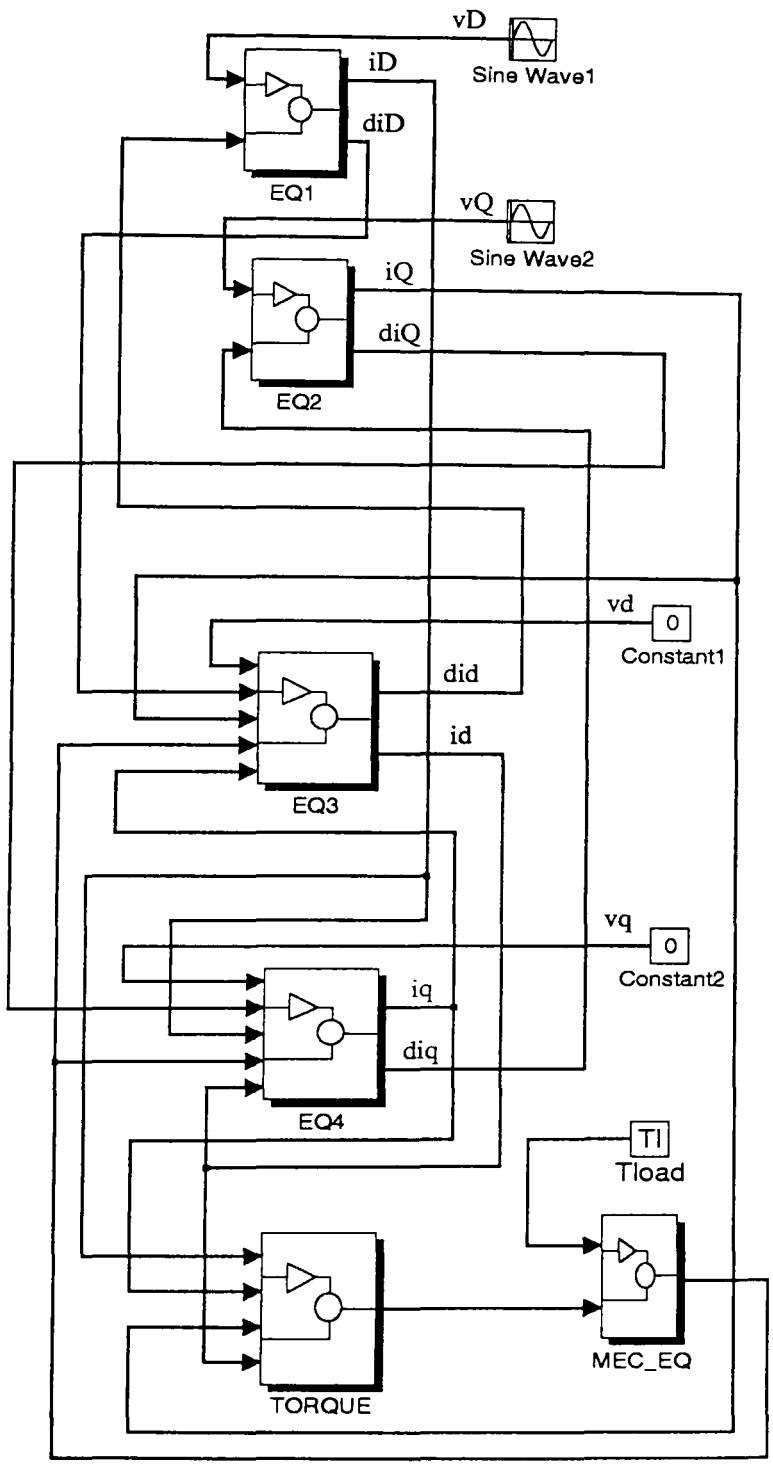

Figure 4.5.1 - The induction motor block diagram

To simplify the display of the overall model, the blocks can be grouped so that the electrical and mechanical systems are *decoupled* as shown in *Figure 4.5.2*. This allows the mechanical system to be linked to the higher harmonics motor models.

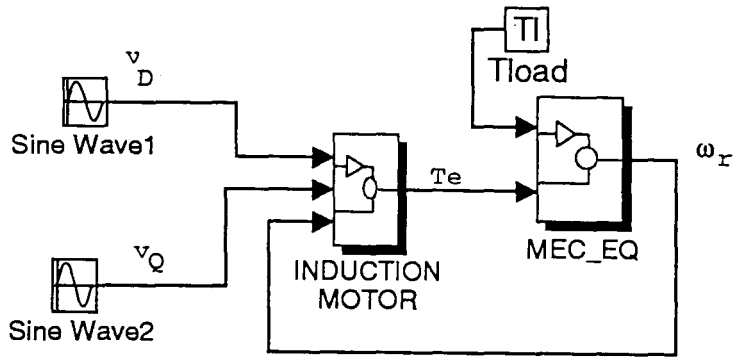

Figure 4.5.2 - Induction motor and the mechanical equation as blocks

For a model utilising the first *7* harmonics of the PWM, the patch diagram shown in *Figure 4.5.3* is obtained.

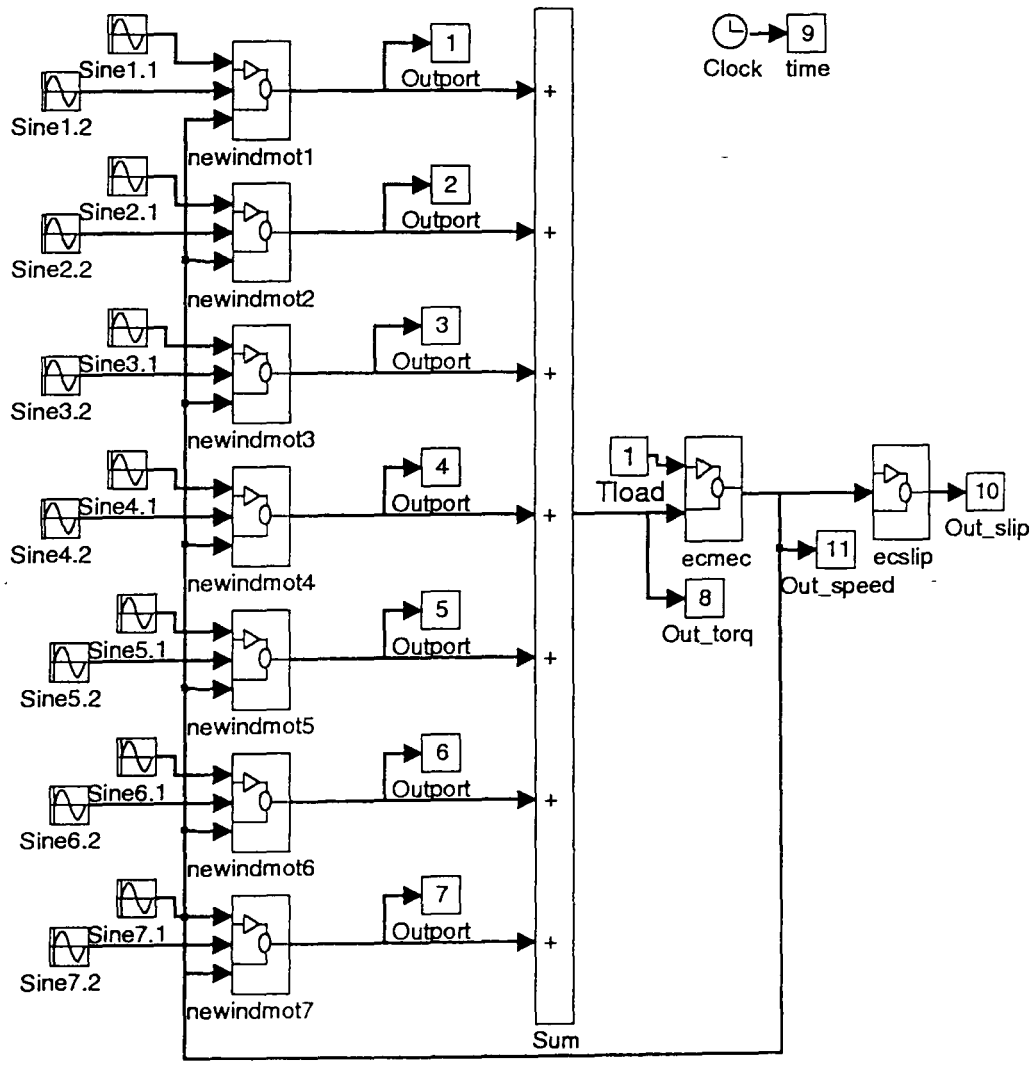

Figure 4.5.3- Seven induction motors assembled to work together

The block diagram in *Figure 4.5.3* is automatically generated by running *mm*, one of the programs specifically written in the MATLAB language by the author. The program *mm* enables a fully functional simulation diagram, corresponding to a user specified harmonic requirement, to be created simply by specifying the harmonics required. Details of *mm* are to be found in Appendix 2.

In order to test the proposed model, simulations are carried out and results plotted and analysed. The Simulink block in *Figure 4.5.3* is simulated for *1.5 seconds*, using the data set from *Section 2.5.*, with the load torque set at *1 Nm*. The results are plotted in the following figures.

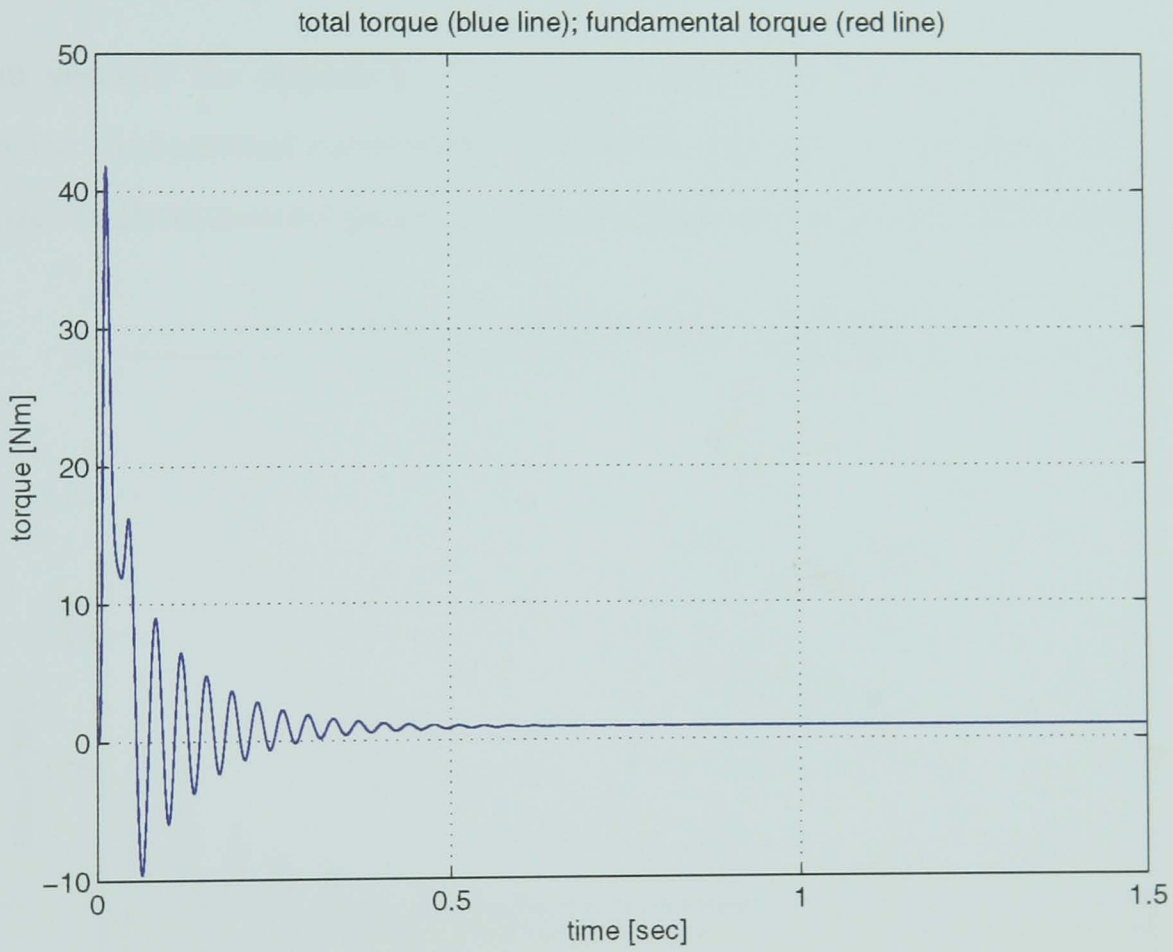

Figure 4.5.4 - Total torque and fundamental torque plotted versus time (7-machine model; Tl=1Nm)

The time-torque graph of *Figure 4.5.4* shows the transient period when the induction motor starts and the steady-state when the torque arrives at the load torque value. Total torque and fundamental torque are plotted together on the same graph in *Figure 4.5.4*, however this is not easily visible. An enlargement of the area between *0.7* and *1.4* seconds and on the *y*-axis between *0.98* and *1.04 Nm* is given in the *Figure 4.5.5*. It is seen that the fundamental is the most important contributor towards the value of the total torque. The percentage of the fundamental torque in the total torque is *99.79 %*. For comparison the contribution in percentage of the other harmonic torques towards the total torque is given in the following table:

harmonic	*fundamental*	3^{rd}	5^{th}	7^{th}	9^{th}	11^{th}	13^{th}
percentage	99.79 %	0 %	0.0001 %	0.0068 %	0.1259 %	0.0492 %	0.0288 %

Table 4.5.1 - Harmonic torques in total torque percentages

Each of the harmonics present in the input voltage waveform produces a rotating m.m.f wave with the same pole number as the fundamental field, but rotating at a higher speed than the synchronous speed. Each rotating m.m.f will produce a torque in the same manner as the fundamental. The harmonic resultant torque is negligible at steady-state and this is proved by the results in *Table 4.5.1.* However additional losses may be introduced by harmonics [84].

This result sustains the approach of analysing and modelling an inverter fed motor by using only the fundamental component. From the percentage values presented in *Table 4.5.1* this approach will not cause significant error in torque calculations in the temporal domain.

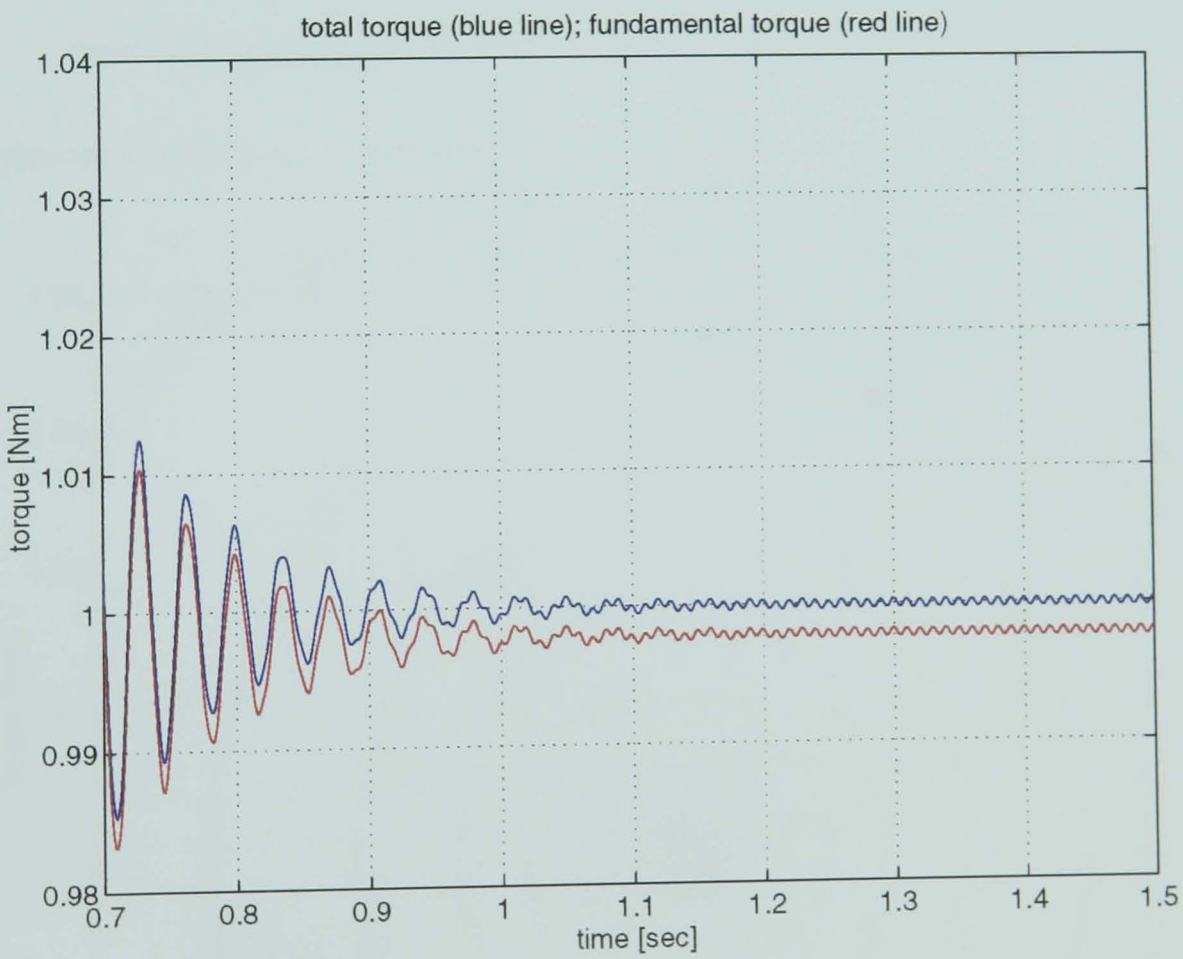

Figure 4.5.5 -Enlargement of Figure 4.5.4 (7-machine model; Tl=1Nm)

Figure 4.5.6 presents the speed-torque plot. It is noticed that the rotor oscillates between values smaller and larger than the synchronous speed until a steady-state is reached at a value around *156.7 rad/s*.

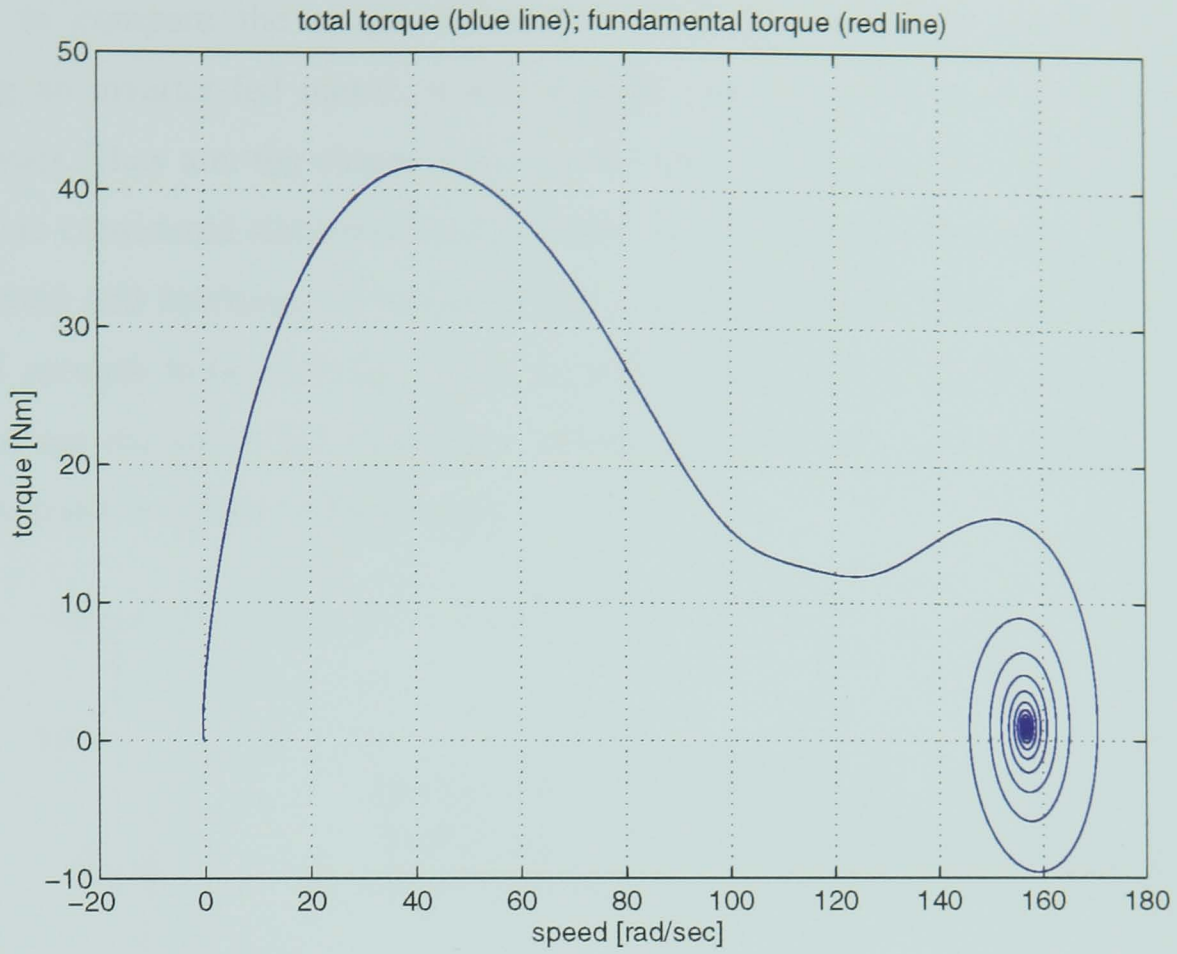

Figure 4.5.6 - The torque plotted versus speed (7-machine model; Tl=1 Nm)

An enlargement of *Figure 4.5.6* is shown in the *Figure 4.5.7*:

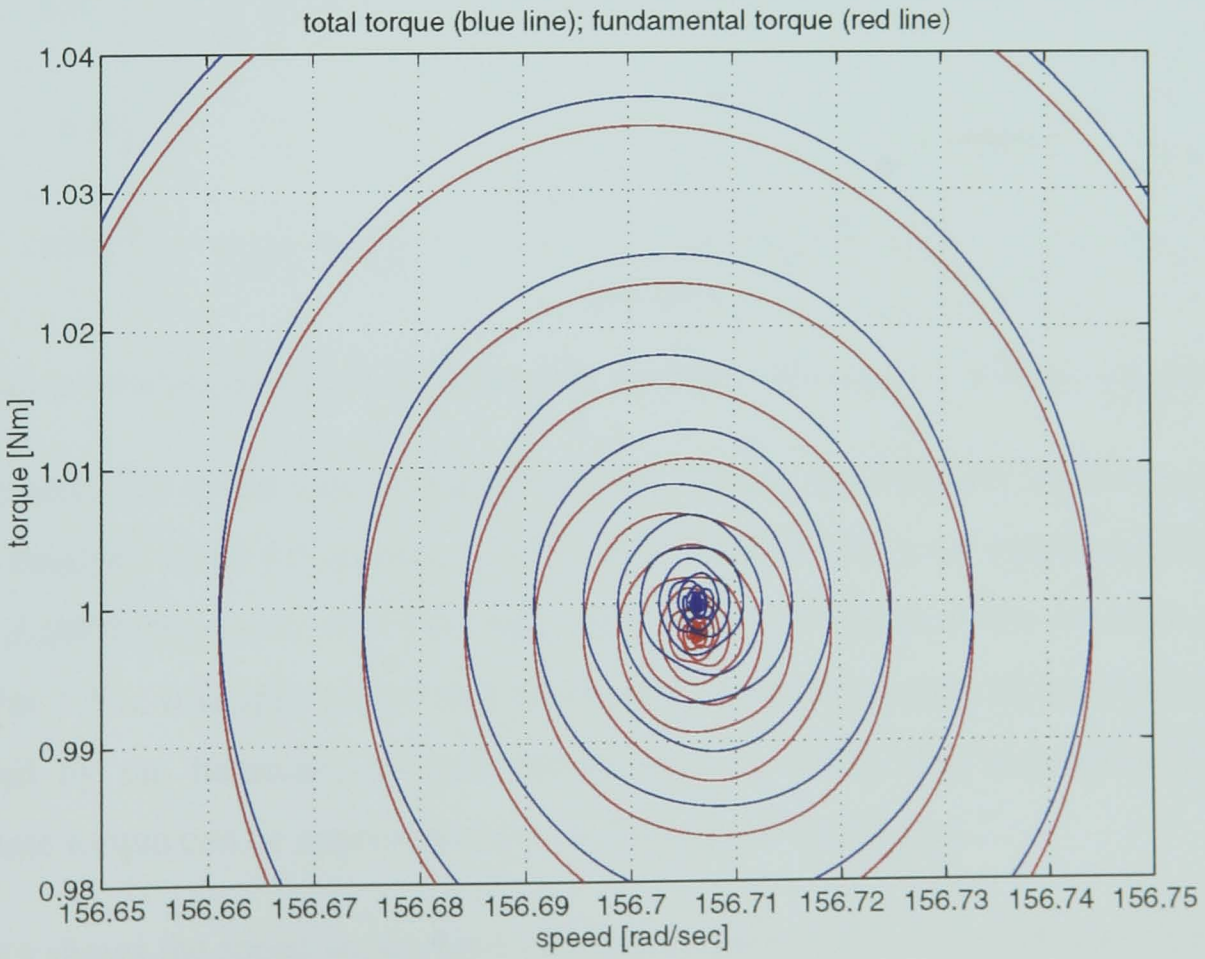

Figure 4.5.7- Enlargement of Figure 4.5.6 (7-machine model; Tl=1Nm)

In order to compare the torque behaviour when the harmonics were considered in modelling an inverter-fed motor, it was thought appropriate to plot on the same graph three torques. They are: the output torque of the one-machine model, when the input stator voltages are considered sinusoidal and therefore the model from *Figure 3.2.2* has been used; also the total and fundamental torques from *Figure 4.5.5*. However this time simulation is run for *2 seconds* to be sure the steady state has been reached. Obviously all simulation conditions are the same: the simulation method, start and stop time and integration step size. This graph is shown in *Figure 4.5.8*.

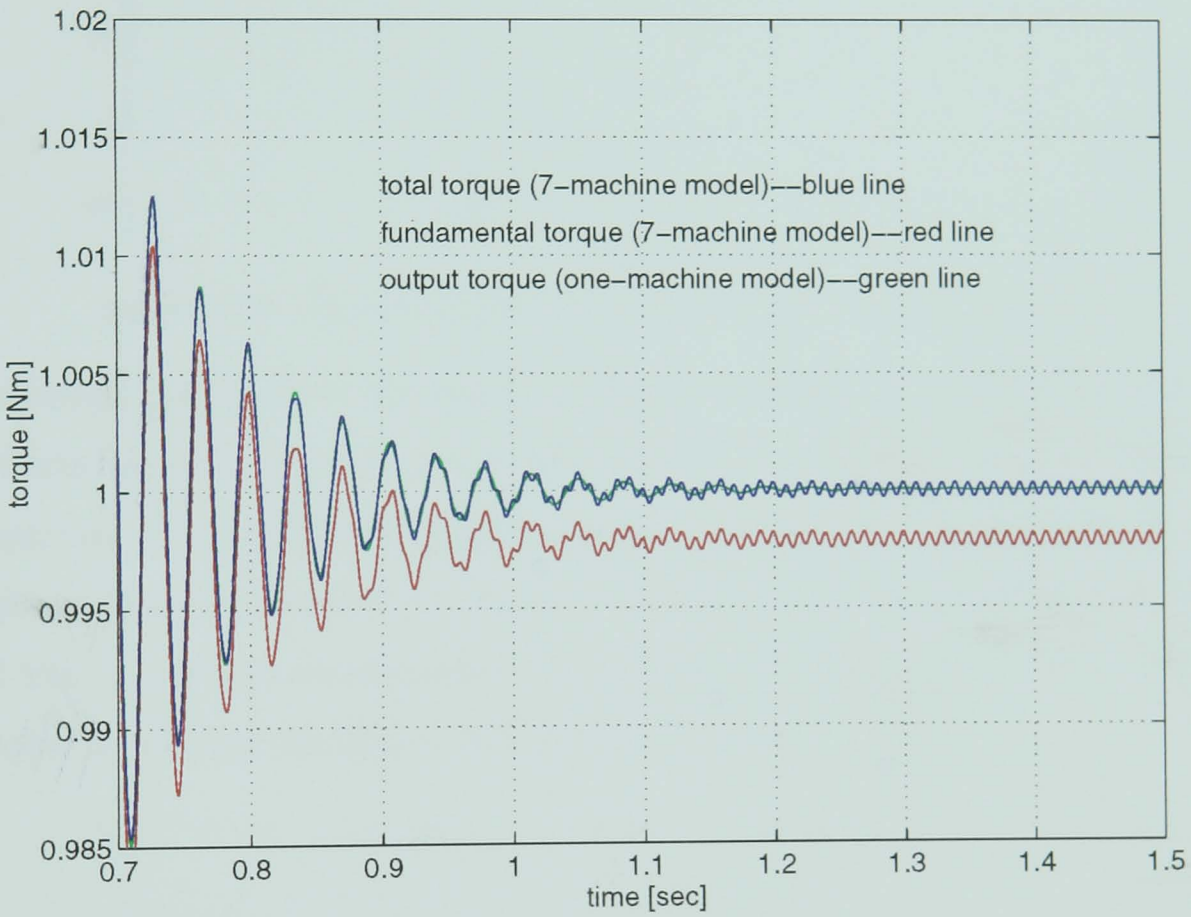

Figure 4.5.8- Comparison between torque/time graphs for one-machine and 7-machine models

It can be seen that in the case of the *7*-machine model, the total and fundamental torque will still oscillate when *1.5 seconds* have passed. However the total torque oscillations are between *1.0002 Nm* and *0.9998 Nm*. The output torque of one-machine model is perfectly constant at *1 Nm*. It is appreciated that the oscillations which appear in the total torque are introduced by the harmonic torques. However their effect may be neglected and the steady-state torque can be approximated to *1 Nm*, the average value.

Figure 4.5.9 shows the speed versus time plot and *Figure 4.5.10* presents the time-slip graph.

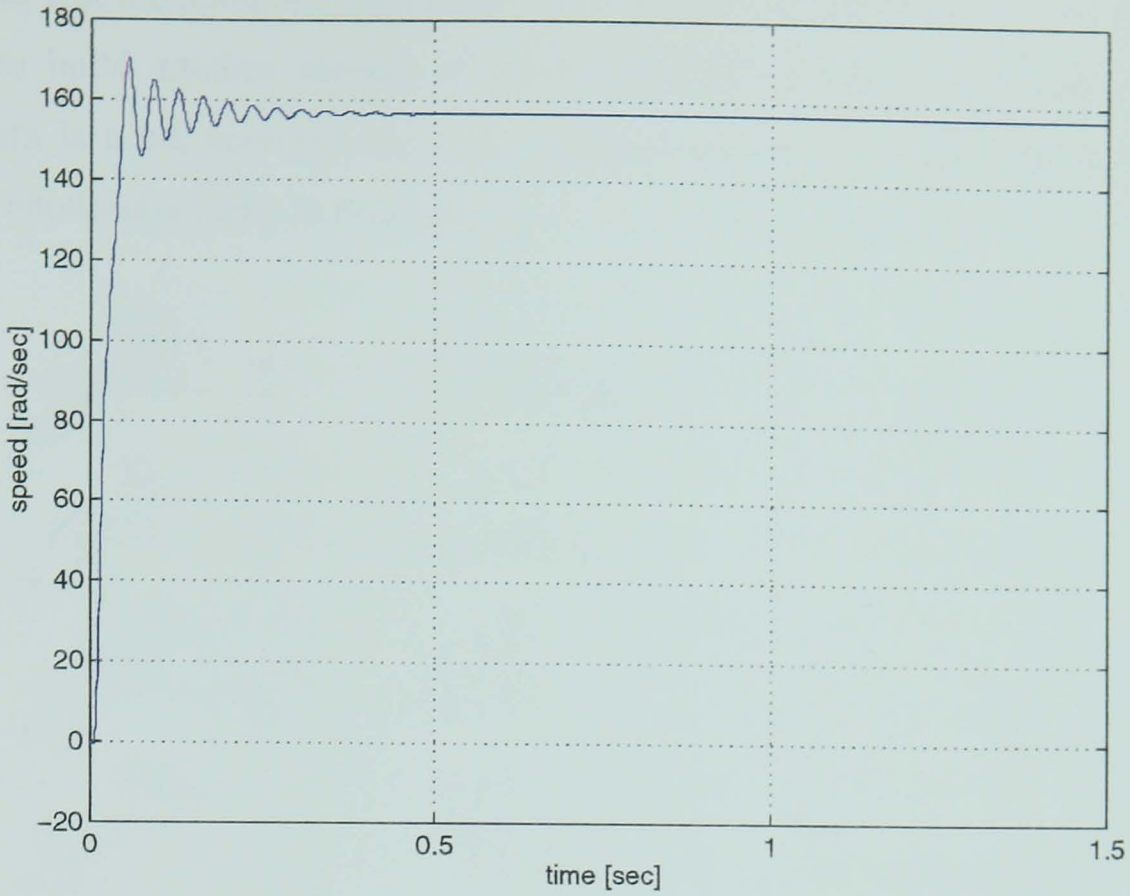

Figure 4.5.9- The speed plotted versus time (7-machine model; Tl=1 Nm)

The time/speed graph shows the speed oscillations inherent in the starting of the motor. These effects are short lived decaying after approximately *0.5 seconds* and by the end of the simulation the rotor has reached the steady-state value of speed. Also the time/slip graph proves that the *Simulink* developed model is correct by showing that the starting value of slip is *1* which corresponds to the starting speed equal to *0*. When the transient period ends, the slip has the steady-state value of *0.0024*.

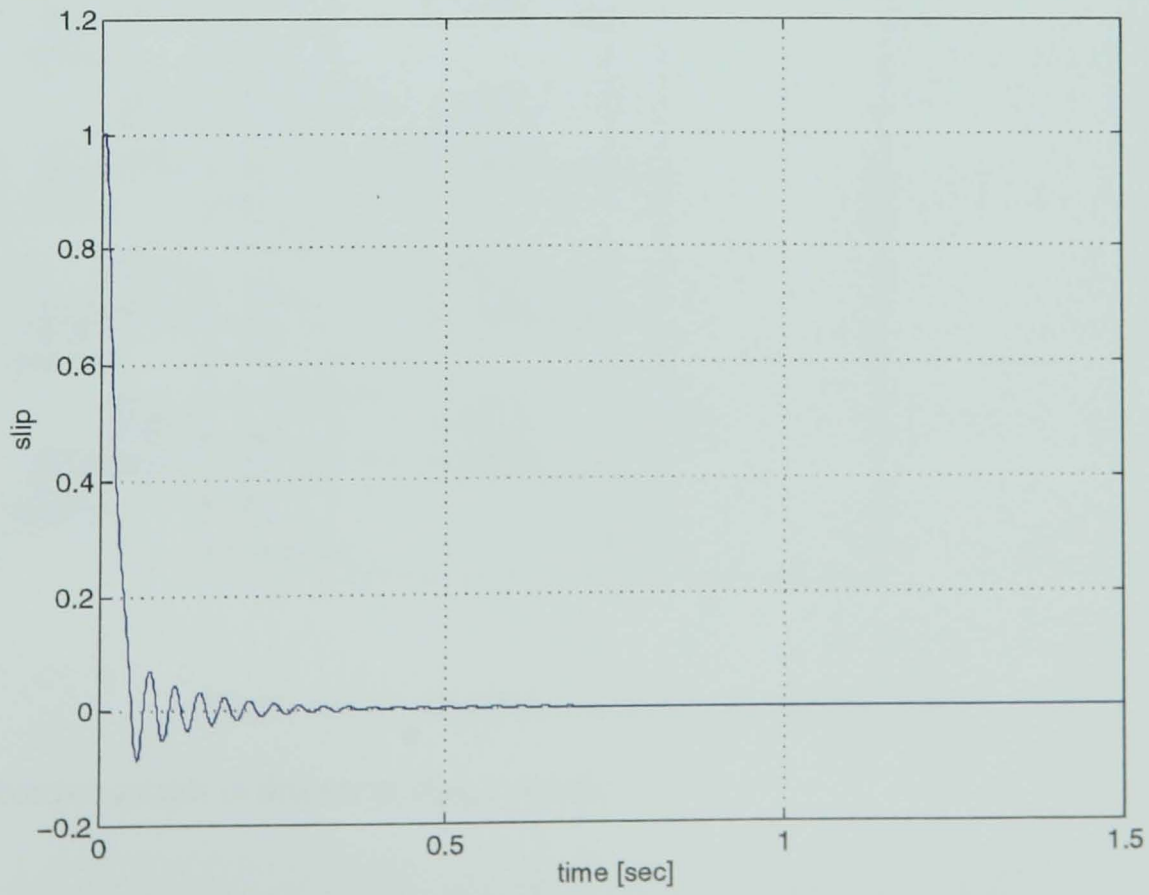

Figure 4.5.10 - The slip plotted versus time (7-machine model; Tl=1 Nm)

To further test the *Simulink* model, the *mm* program is run and the first *11* harmonics are chosen to build another version of the multi-machine model. The same set of motor parameters is used, however the load torque is imposed to *5 Nm.* The Simulink patch diagram obtained is given in *Figure 4.5.11.*

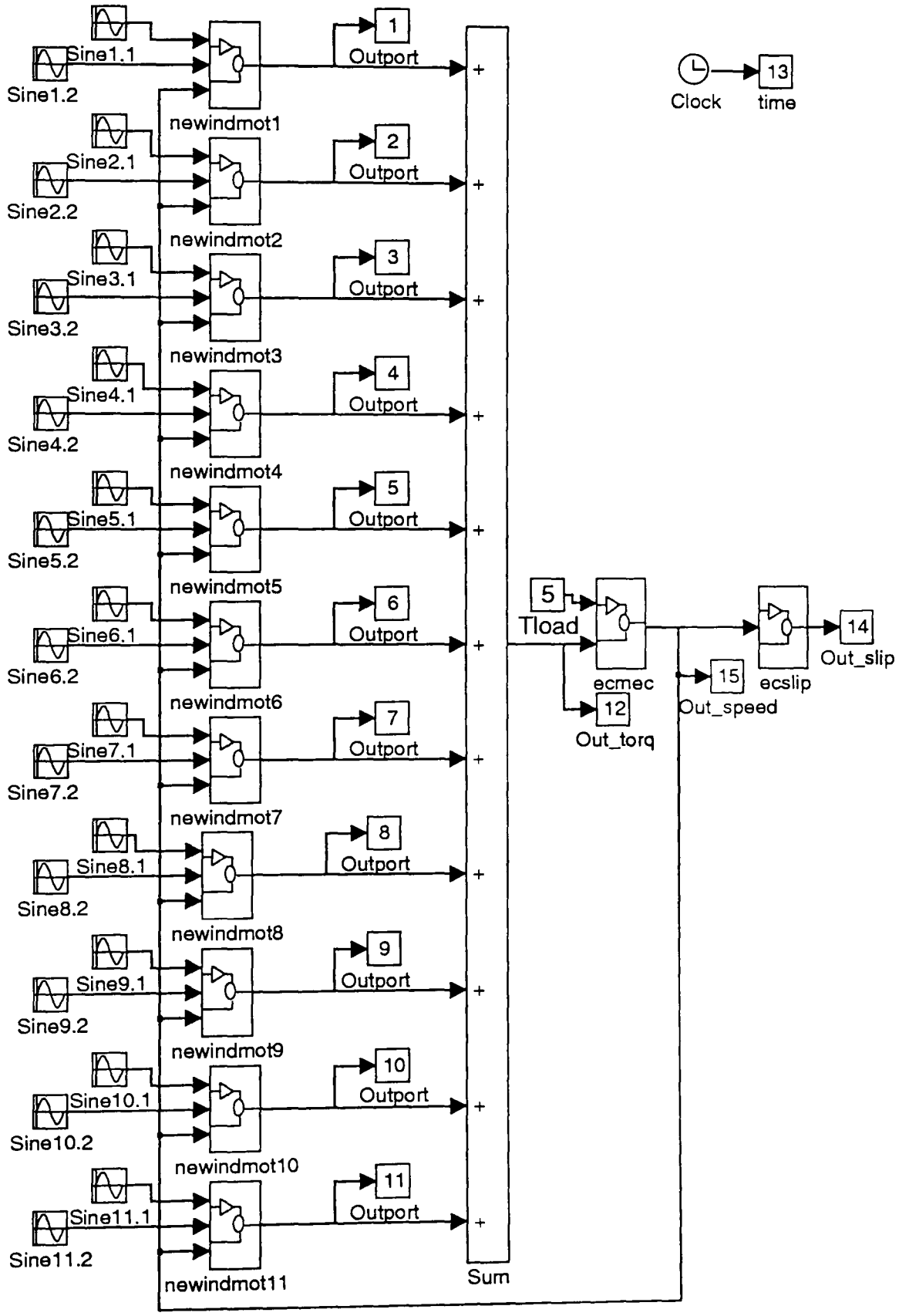

Figure 4.5.11 - Simulink diagram for 11-machine model

The time/torque graph is drawn in *Figure 4.5.12.*

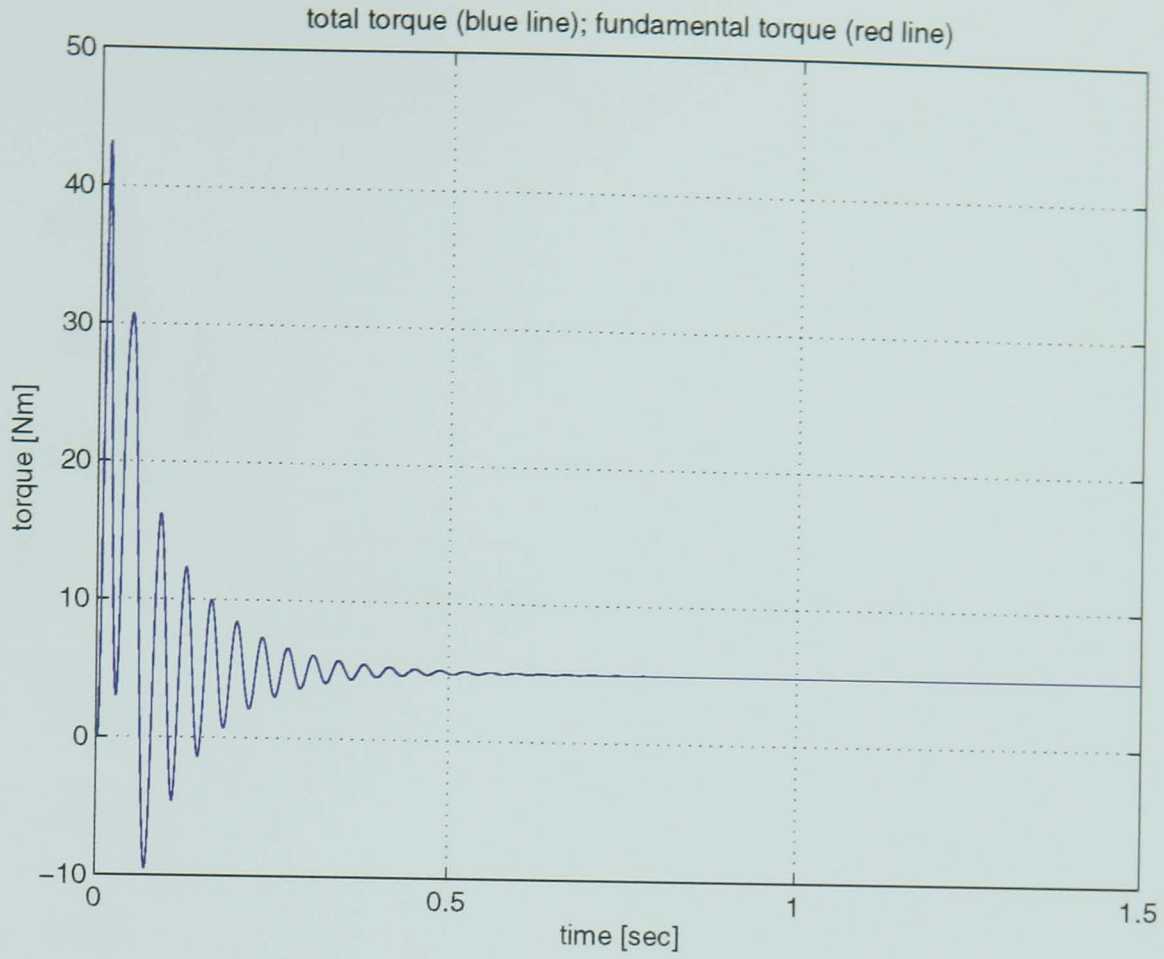

Figure 4.5.12 - Total torque and fundamental torque plotted versus time(11-machine model; Tl=5Nm)

To identify the fundamental and total torque, an enlargement of *Figure 4.5.12* is given in *Figure 4.5.13.* In *Table 4.5.2* amplitude and frequency information are given about the harmonics employed in the *11*-machine model [*Figure 4.5.11*]. Also, last column shows percentages of harmonic torques in the total torque. The percentage of fundamental torque into the total torque is *99.95 %*. It can be seen that the 19^{th} and 21^{st} harmonics produce negative torques. This means that the particular harmonics produce currents with inverse sequence with respect to the fundamental. Also it is noted that the harmonics having the amplitude of about one fifth of the fundamental amplitude give most notable contributions to the total torque.

harmonic number	***harmonic amplitude [V]***	***frequency [rad/sec]***	***percentage [V]***
fundamental	*508.2691*	*314.16*	*99.95*
3^{rd}	*-0.0404*	*942.48*	*0*
5^{th}	*-1.1067*	*1570.79*	*0*
7^{th}	*-16.8716*	*2199.11*	*0.0014*
9^{th}	*-107.9006*	*2827.43*	*0.0252*
11^{th}	*-92.1310*	*3455.75*	*0.0099*
13^{th}	*91.5850*	*4084.07*	*0.0058*
15^{th}	*103.1867*	*4712.39*	*0.0045*
17^{th}	*-8.6151*	*5340.71*	*0*
19^{th}	*-59.2351*	*5969.03*	*-0.0245*
21^{st}	*-4.9120*	*6597.34*	*-0.0009*

Table 4.5.2 - Harmonic torques in total torque percentages (11-machine model, Tl=5 Nm)

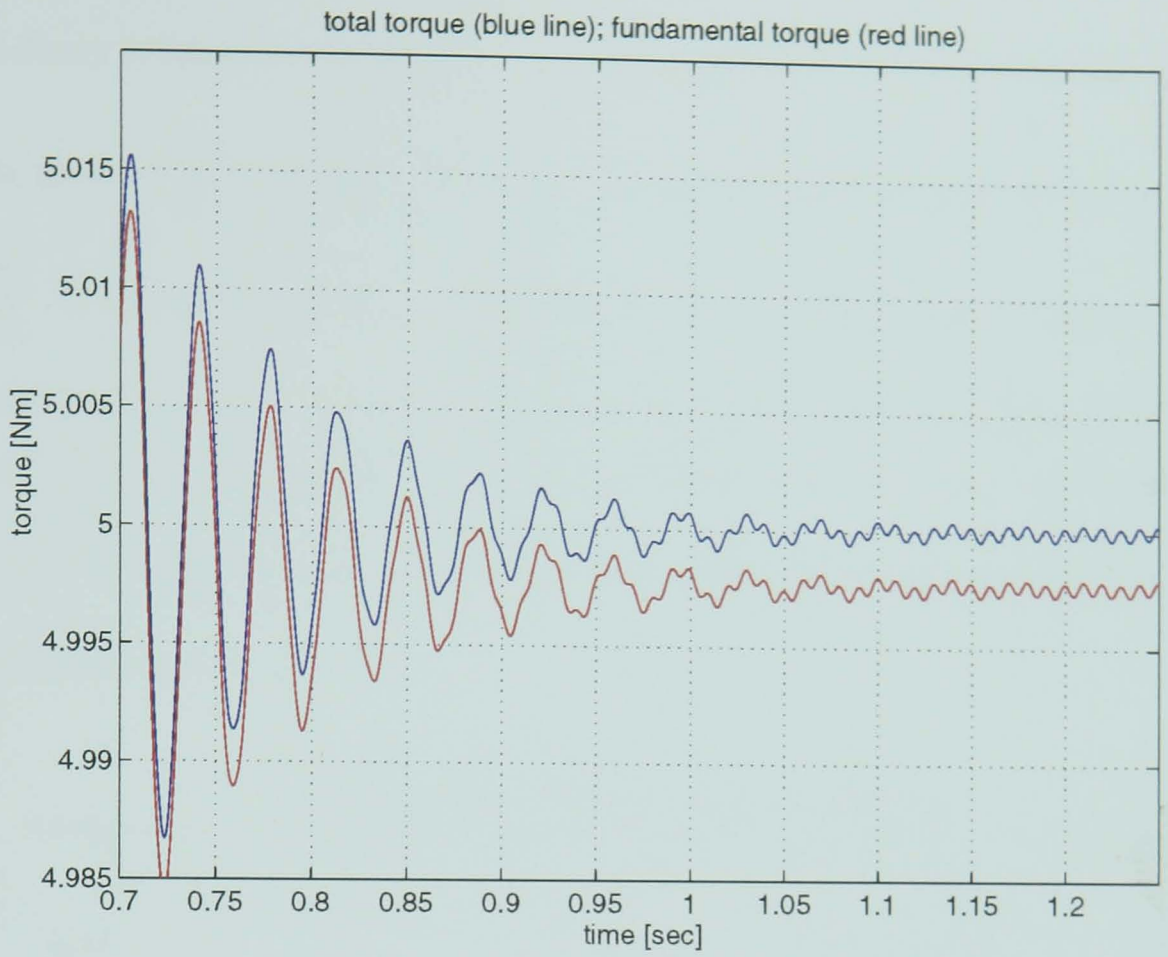

Figure 4.5.13 - Enlargement of Figure 4.5.12 (11-machine model, Tl=5 Nm)

The speed/torque graph is shown in *Figure 4.5.14* and subsequently an enlargement is given in *Figure 4.5.15*. The enlargement permits identifying the total and fundamental torque and the value of the steady-state speed as *155.155* rad/sec.

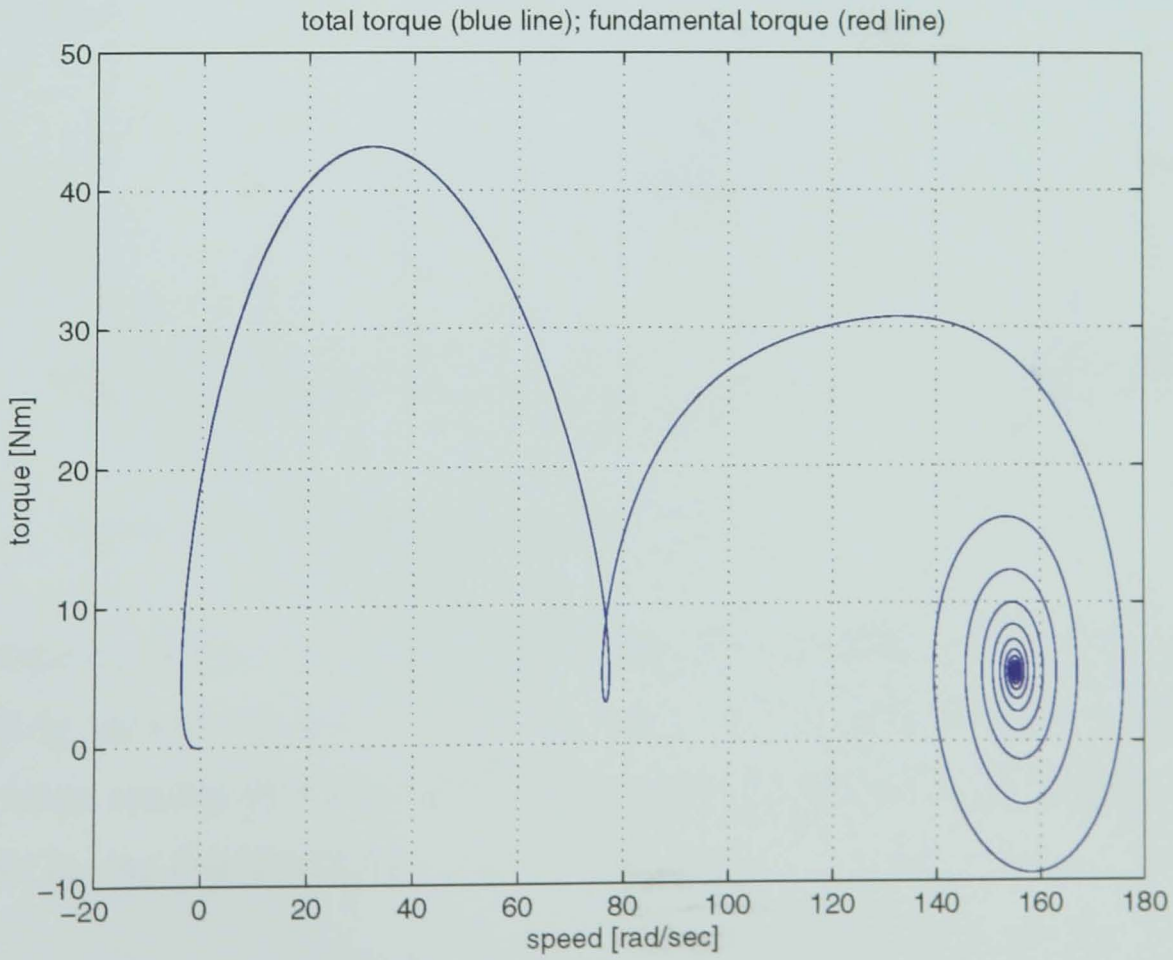

Figure 4.5.14 -Speed/Torque graph (11-machine model, Tl=5 Nm)

Immediately after the start of the drive, the torque will reach a maximum of *~43 Nm* and then will decay towards *2 Nm*. From $T = J\frac{d\omega}{dt} + D\omega + T_{load}$ and knowing that T_{load}=*5 Nm* and D= 0, then: $J\frac{d\omega}{dt} = T - T_{load}$. Therefore when torque goes smaller than the load torque then $J\frac{d\omega}{dt}$ is negative and speed will decrease. This is illustrated in *Figure 4.5.14*. Then torque increases again and speed will follow in the same manner. Reaching the region *140-180 rad/s*, rotor speed oscillates between under synchronous and over synchronous speed values before stabilising. Every time when torque becomes smaller than *5 Nm*, $J\frac{d\omega}{dt}$ is negative and speed decreases.

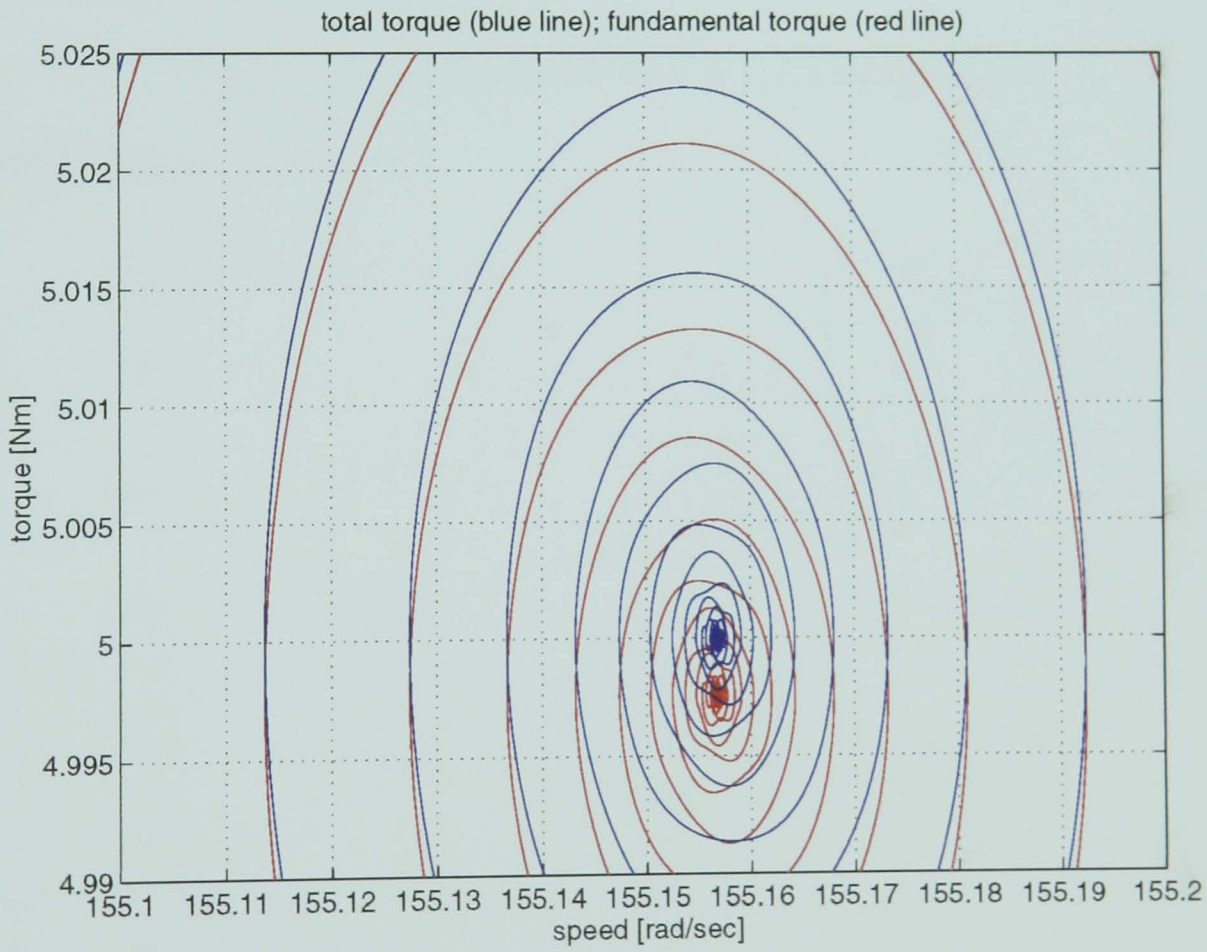

Figure 4.5.15 - Enlargement of Figure 4.5.14 (11-machine model, Tl=5 Nm)

The time/speed graph in *Figure 4.5.16* shows the transient period of the first *0.5 seconds* and settling of the speed towards the steady-state value. Also it shows the speed having a slight decrease when reaches *80 rad/sec*. This corresponds in *Figure 4.5.14* to torque declining to *3 Nm* after having reached *43 Nm* .

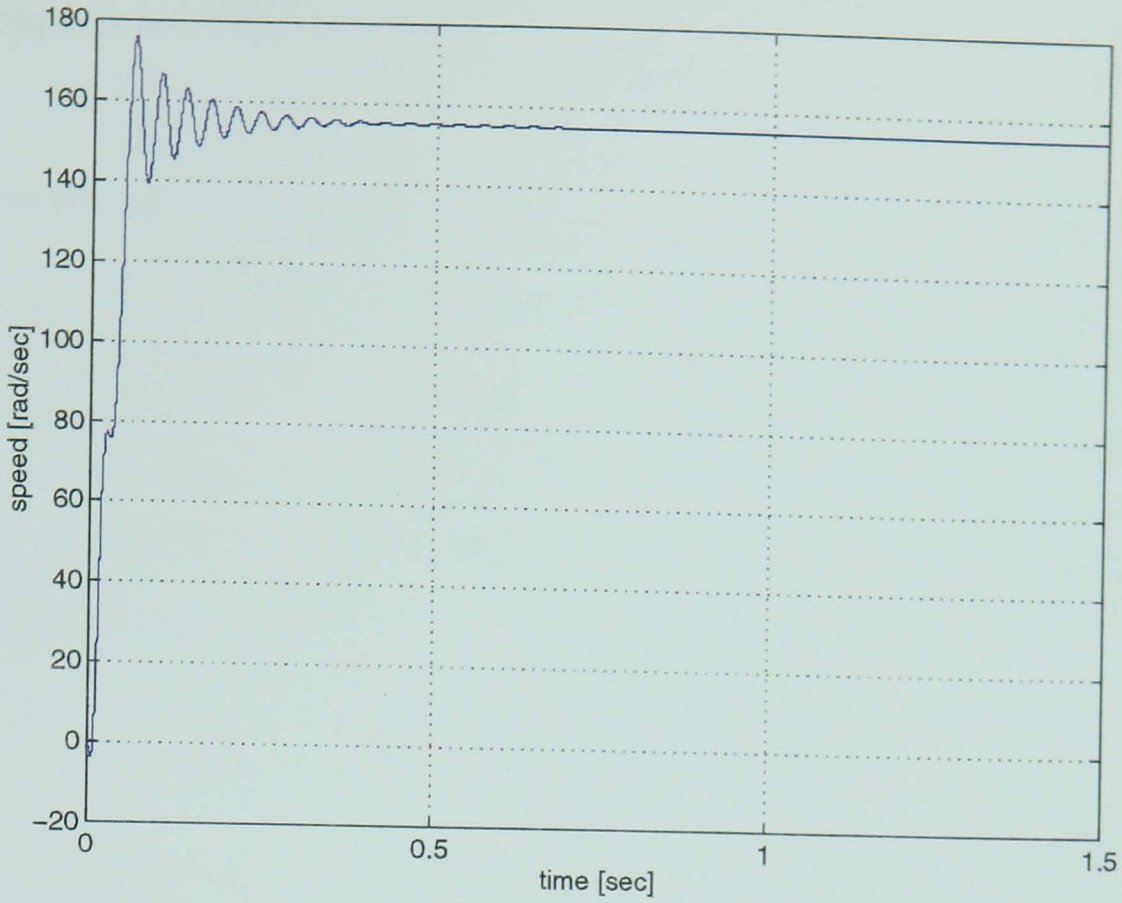

Figure 4.5.16 -Time/Speed graph (11-machine model; Tl=5 Nm)

The time/slip graph:

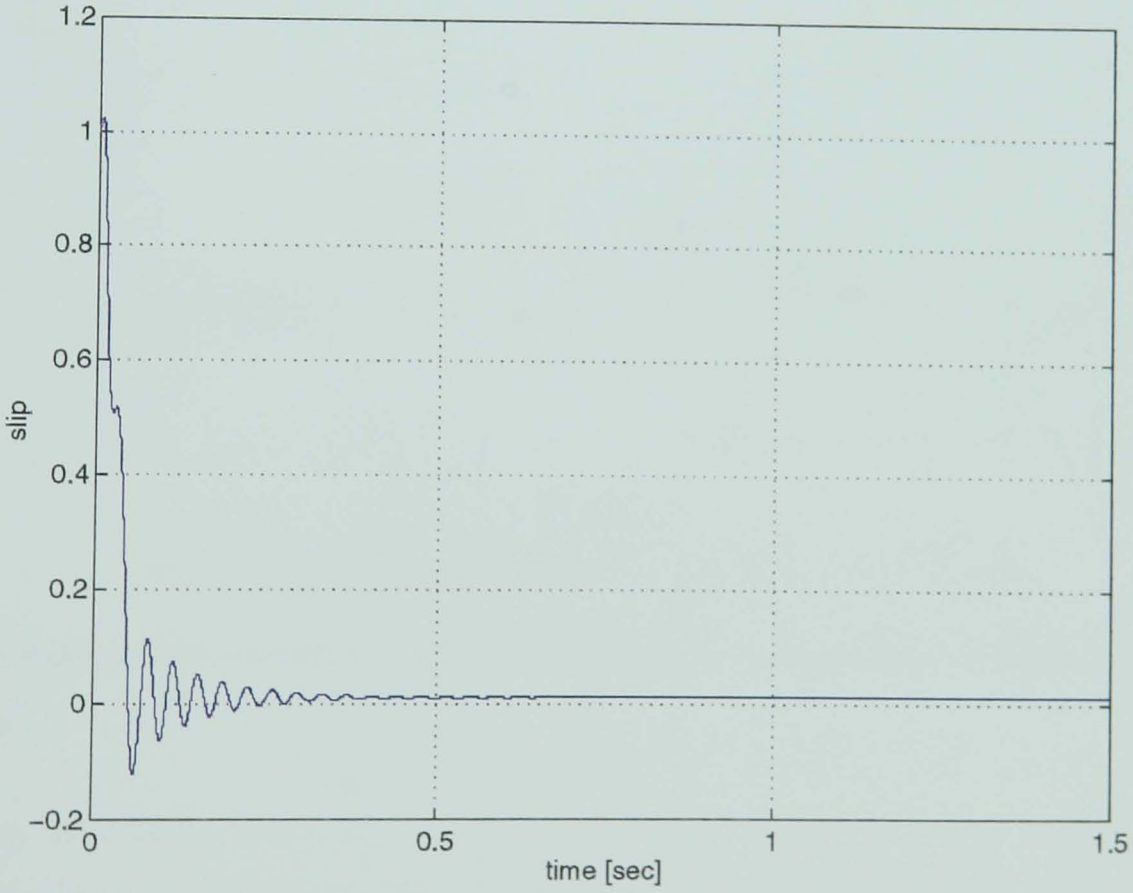

Figure 4.5.17 -Time/Slip graph (11-machine model; Tl=5 Nm)

Two examples of the multi-machine model were given to prove the functionality of the model designed by the *mm* program. In all previous simulations, the load torque has been considered as a constant equal to values from *0.1 Nm* to *5 Nm*. As in practical situations

the load torque is not always constant quantity, it was considered appropriate to simulate a load step.

The next results will illustrate the response of the motor model to the simulation of a load step. This is carried out on the *11*-machine model. For this purpose, the `*Constant*` block which simulates the load torque [*Figure 4.5.11*] is replaced by a `*Step Input*` block. The load torque is imposed as *1 Nm* at start and then will step to *5 Nm* when simulation time exceeds *1 second*. Simulation is carried out for *1.5 seconds* and simulation results are presented as follows. In *Figure 4.5.18* , the total torque is plotted together on the same graph with the imposed load torque.

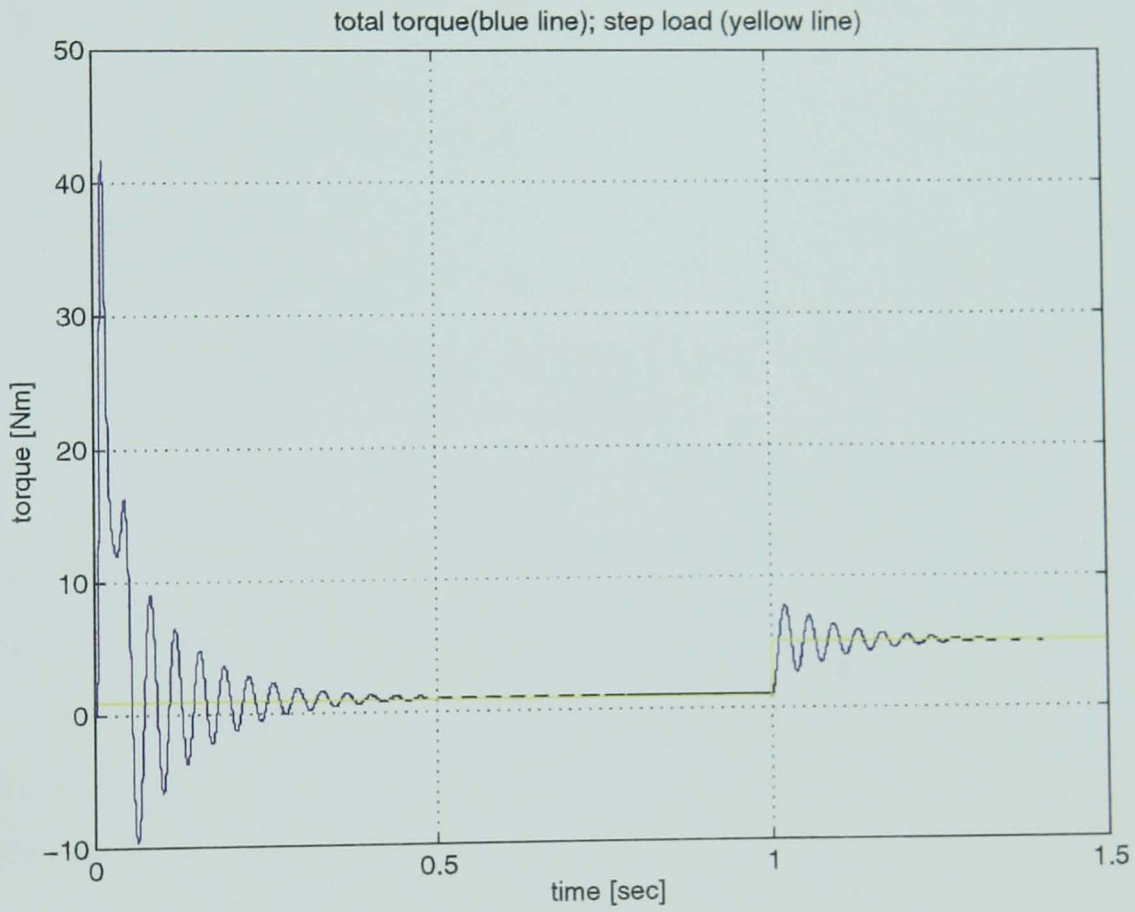

Figure 4.5.18 -Time/Torque graph (11-machine model; step load)

The time/torque graph of *Figure 4.5.18* proves that the motor torque follows the reference load torque.

Steady-state is reached and torque is around *1 Nm* when simulation time arrives at *1 second*. The load step causes a new transient period and after the oscillations have diminished, torque has reached *5 Nm*. From *Figure 4.5.19* the speed is seen to decrease from *156.69 rad/sec* to *155.16 rad/sec* when steady state is reached for the second time. This shows that the motor model is capable to track and follow the load torque, however as the supply has been kept constant all simulation time, an increase in the required level of

torque causes the motor to lose its level of speed. This can be seen in *Figure 4.5.19* where the time/speed graph is plotted, again proving that the drive model behaves correctly and the speed response is stable and as expected.

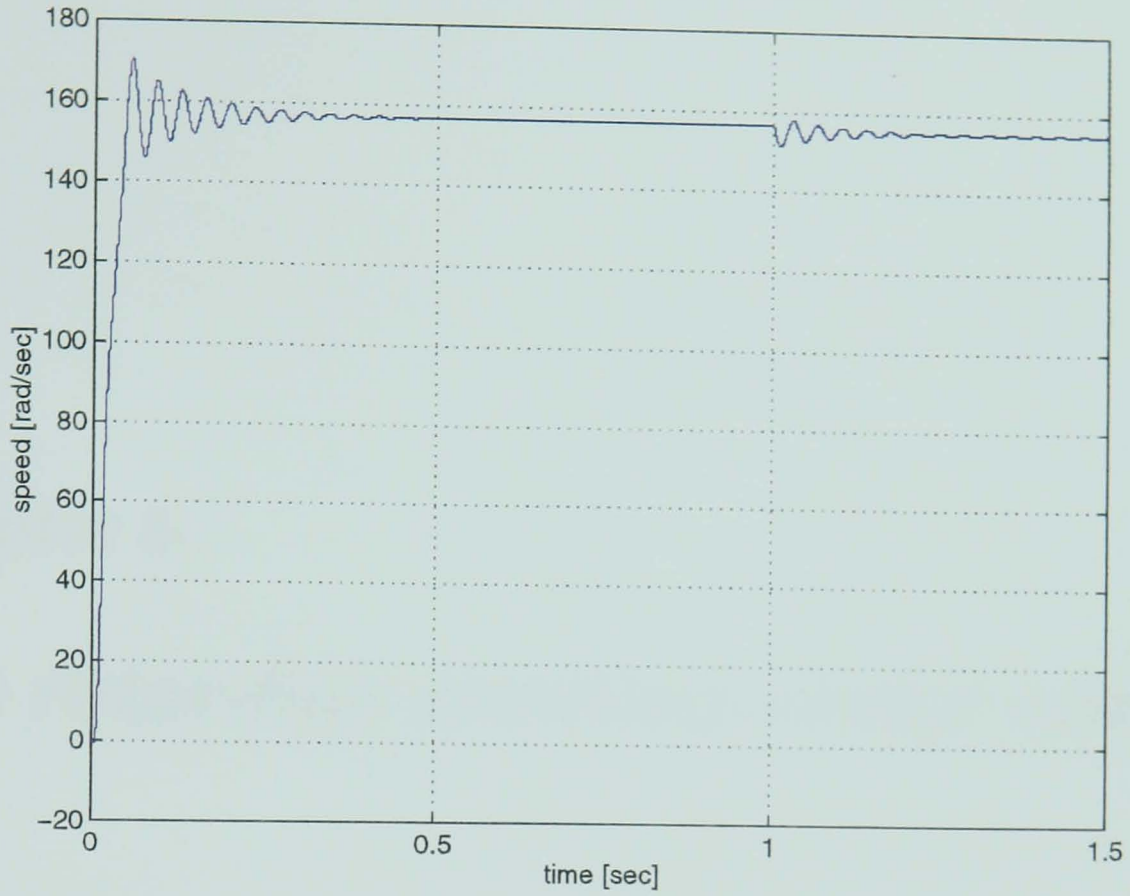

Figure 4.5.19 -Time/Speed graph (11-machine model; (step load)

In this section a model for an inverter fed motor has been developed and presented. Two test models produced using a different content of harmonics in each case are shown and simulated. The second model, the *11*-machine model was also tested by applying a step load torque. Results show a good adaptation of the motor to the required level of torque with a detriment of reduction in speed as no feed-back control is employed.

Chapter 4 presents a novel modelling approach for an inverter-fed induction motor, realised with the Simulink/MATLAB toolbox. The model can be used by running a program written by the author in the MATLAB language. This program allows the user to choose a certain harmonic content of the PWM waveform supplying the induction motor. Various simulations have been run and reported results show very good behaviour of the model which supports a general confidence level with respect to results presented.

The next chapter presents the theoretical basis of the vector control theory. Starting with the general approach to space vector theory, it continues by presenting the equations for rotor-flux oriented control.

Chapter 5

The rotor-flux oriented vector control

As described in *Section 1.2*, the objective of vector control is to control independently the two components of current, i.e. the component producing torque and that producing flux. This is not easily achieved because the two components of current are supplied via the same three-phase stator windings. The solution is to reference the voltages and currents to a special rotating reference frame which could be fixed either to the rotor flux space phasor, the stator flux space phasor or the magnetising flux space phasor [68]. Each of these cases result in a specific vector control strategy but there are inherent similarities between the strategies.

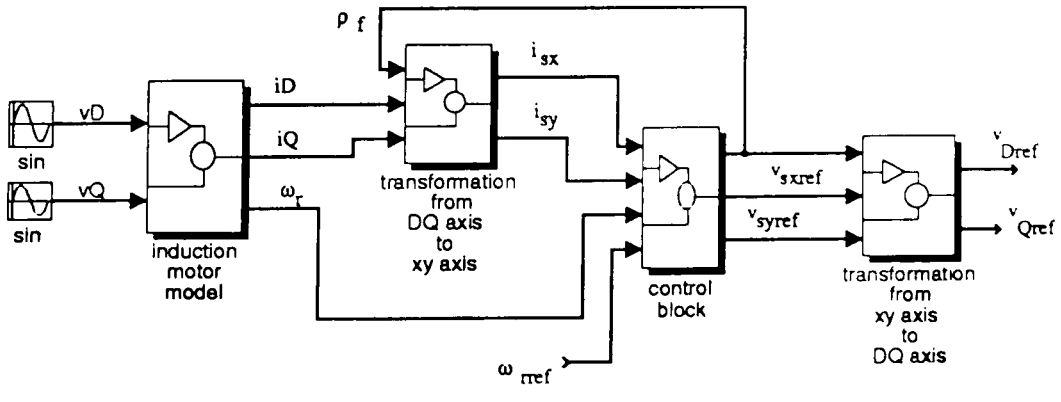

Figure 5.1 - A vector control scheme

Figure 5.1 shows the principle of a vector controlled scheme for an induction motor. It can be seen that the *DQ*-axis induction motor model forms part of the scheme where the required inputs are the *d* and *q*-axis voltages and the outputs are the stator currents and the rotor speed. The basis of this speed control scheme is that balanced 3-phase voltages are initially transformed, physically to the direct and quadrature voltages, v_D and v_Q respectively. Further transformation of v_D and v_Q, using the machine parameters, yields the *DQ*-axis currents. The key to this scheme lies in the way these currents are converted to 3-phase voltages or firing signals to the power electronic circuitry. For this closed loop system the actual speed of the motor is compared against the required reference speed and the difference is used to condition the voltages v_{sxref} and v_{syref}.

This chapter deals with the basis of rotor-flux oriented control. It starts with a section about the space phasor theory followed by the stator and rotor equations using the space vectors. Furthermore a general reference frame, independent of the stator or the rotor frames is defined. The voltage equations written in this general frame will characterise the rotor-flux control when the general frame becomes a particular case by aligning it to the rotor-flux space vector.

5.1 The Space Phasor Theory - Basics

Based on *DQ*-axis theory a model for the induction motor has been developed. From this model different mathematical quantities such as stator currents: i_D and i_Q, rotor currents i_d and i_q, can be obtained. The direct and quadrature currents are fictitious currents which relate to the actual 3-phase currents according to *(2.5.1)*:

$$i_D = c(i_A - \frac{1}{2} i_B - \frac{1}{2} i_C) \qquad (5.1.1)$$

$$i_Q = c\frac{\sqrt{3}}{2}(i_B - i_C)$$

If the transformation from *3*-phase to *dq*-axis is considered a magnitude invariant transformation then *c* is taken as $(2/3)$. This has the advantage of preserving the magnitudes across the transformation. The power invariant form obviously maintains the power invariant across transformation and uses the constant c as $\sqrt{(2/3)}$.

A space phasor for the stator currents can now be defined using the DQ-axis currents i_D and i_Q, i.e.:

$$\bar{i}_s = i_D(t) + j\, i_Q(t) \qquad (5.1.2)$$

where: $\bar{i}_s$ is the space phasor of the stator currents,

$i_D(t)$ is the instantaneous value of the D-axis stator current component,

$i_Q(t)$ is the instantaneous value of the Q-axis stator current component.

In a short notation $\bar{i}_s$ can be expressed as:

$$\bar{i}_S = |\bar{i}_S| e^{j\alpha_S} \qquad (5.1.3)$$

where: $|\bar{i}_S|$ is the modulus of the space phasor $\bar{i}_S$,

α_S is the phase angle with the D-axis.

To better visualise the space vector $\bar{i}_s$, a phasor diagram is shown in *Figure 5.1.1*. The rotor frame is noted ($\alpha\beta$) axes and the stationary frame fixed to the stator is given by (DQ) axes. The angle between the two frames is the rotor angle, θ_r.

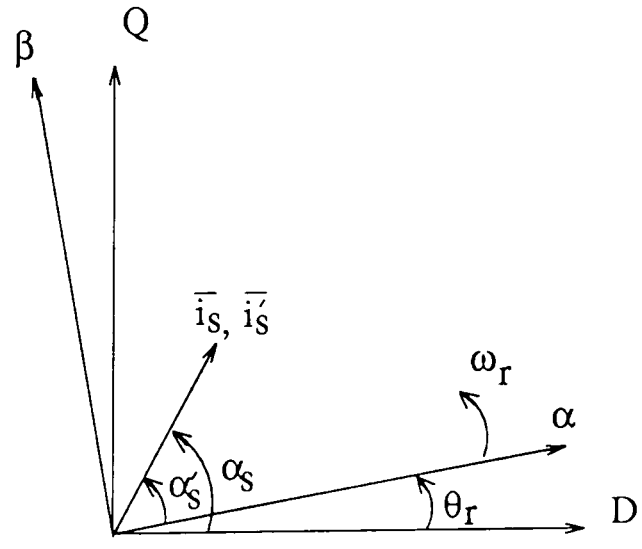

Figure 5.1.1 - The space phasor of the stator currents

The stator current phasor may be also expressed in ($\alpha\beta$) frame by transforming the phase angle, as follows:

$$\bar{i}'_S = |\bar{i}_S| e^{j\alpha'_S} = |\bar{i}_S| e^{j(\alpha_S - \theta_r)} = \bar{i}_S e^{-j\theta_r} \qquad (5.1.4)$$

Similarly a space phasor for the rotor currents is $\bar{i}_r$, can be defined as:

$$\bar{i}_r = i_{r\alpha} + j\, i_{r\beta} \qquad \text{or shorter:} \qquad (5.1.5)$$

$$\bar{i}_r = |\bar{i}_r| e^{j\alpha_r} \qquad (5.1.6)$$

and can be seen in the next figure:

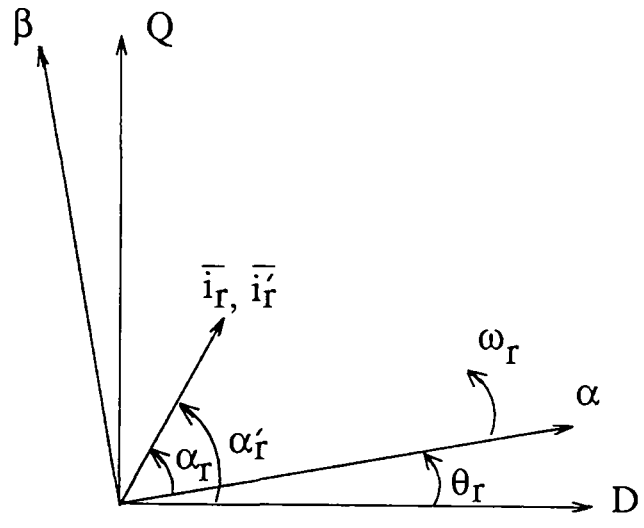

Figure 5.1.2 The space phasor of the rotor currents

In equation *(5.1.6)* $\bar{i}_r$ is expressed in the reference frame fixed to the rotor as shown in *Figure 5.1.2.* Transformation to the stationary frame yields:

$$\bar{i}'_r = |\bar{i}_r| e^{j\alpha'_r} = |\bar{i}_r| e^{j(\alpha_r + \theta_r)} = \bar{i}_r e^{j\theta_r} \qquad (5.1.7)$$

where $\bar{i}'_r$ is the space phasor of the rotor currents (in the stator frame).

Due to the combined effects of the stator and rotor m.m.f.s, the resultant wave will be their sum and therefore the magnetising current space phasor , $\bar{i}_m$ is introduced as being the sum of the stator current space phasor and the rotor current space phasor expressed in the stator reference frame. The rotor current is referred to the stator.

$$\bar{i}_m = \bar{i}_s + \frac{N_2 k_{w2}}{N_1 k_{w1}} \bar{i}'_r \qquad (5.1.8)$$

where: N_1 is the number of turns in stator winding,

k_{w1} is the winding factor of the stator winding,

N_2 is the number of turns in rotor winding and

k_{w2} is the winding factor of the rotor winding.

The stator flux linkage space phasor $\overline{\psi}_s$ is now introduced considering that there are two components making up the flux: the first is the self flux space phasor produced by the stator currents and the second is the mutual flux linkage space phasor, due to the rotor currents. Thus its expression in the stationary reference frame is:

$$\overline{\psi}_S = L_S \bar{i}_S + L_m \bar{i}_r e^{j\theta_r} \qquad (5.1.9)$$

where: L_s is the so-called total 3-phase stator inductance and

L_m is the 3-phase magnetising inductance.

Expressed in the reference frame fixed to the rotor, $\overline{\psi}_s$ becomes:

$$\overline{\psi}'_S = L_S \bar{i}'_S + L_m \bar{i}_r \qquad (5.1.10)$$

The rotor flux linkage $\overline{\psi}_r$ is given by:

$$\overline{\psi}'_r = L_r \bar{i}'_r + L_m \bar{i}_S \qquad (5.1.11)$$

and in the frame fixed to the rotor:

$$\overline{\psi}_r = L_r \bar{i}_r + L_m \bar{i}'_s \qquad (5.1.12)$$

The space vector notion has been introduced in this section and current and flux linkages space vectors have been given in the rotor and stator frame. Next, the voltage equations are written using the previously defined space vectors.

5.2 The stator and rotor equations

As different space phasors are available, the space phasor form of the voltage equations can now be written down as:

$$\bar{v}_S = R_S \bar{i}_S + \frac{d\overline{\psi}_S}{dt} \qquad (5.2.1)$$

for the stator and for the rotor (in the stator reference frame):

$$\bar{v}'_r = R_r \bar{i}'_r + \frac{d\overline{\psi}'_r}{dt} - j\omega_r \overline{\psi}'_r \qquad (5.2.2)$$

where: $\bar{v}_s$ is the stator voltage space vector and

$\bar{v}_r'$ is the rotor voltage space vector.

By replacing $\bar{\Psi}_S$ and $\bar{\Psi}'_r$ using equations *(5.1.9)* and *(5.1.11)* , *(5.2.1)* and *(5.2.2)* become:

$$\bar{v}_S = R_S\bar{i}_S + \frac{d(L_S\bar{i}_S)}{dt} + \frac{d(L_m\bar{i}'_r)}{dt}$$

$$\bar{v}'_r = R_r\bar{i}'_r + \frac{d(L_r\bar{i}'_r)}{dt} + \frac{d(L_m\bar{i}_S)}{dt} - j\omega_r(L_r\bar{i}'_r + L_m\bar{i}_S) \qquad (5.2.3)$$

These equations are written in the stationary frame fixed to the stator. However to be able to achieve vector control, all quantities are expressed in a different frame.

Initially, it can be a general frame which may be then assigned to any of the desired chosen flux phasors. This general reference frame is introduced in *Section 5.4* after a few considerations about torque are given in the subsequent section.

5.3 The electromagnetic torque

In a *d.c.* drive, good torque and speed response can be achieved. Torque is controlled using the armature and field current as the expression of torque, t_e, in a *d.c.* machine is given by:

$$t_e = c\bar{\Psi}_f \times \bar{i}_a \qquad (5.3.1)$$

where: $\bar{\Psi}_f$ the flux linkage space phasor produced by the field current,

$\bar{i}_a$ is the space phasor of the armature current,

c is the machine constant.

By analogy to *(5.3.1)*, the torque produced in an induction machine is defined as:

$$t_e = c\bar{\Psi}_S \times \bar{i}'_r \qquad (5.3.2)$$

where $\bar{\Psi}_s$ and $\bar{i}_r'$ are the space phasors described in the previous section and c is a constant. Linear magnetic conditions are assumed and therefore c is taken as a constant.

In a scalar form equation *(5.3.2)* becomes:

$$t_e = c\left|\bar{\Psi}_s\right|\left|\bar{i}_r\right|\sin\gamma \qquad (5.3.3)$$

where γ is the torque angle and maximum of torque is obtained at $\gamma=90°$. This equation can be also proved on energy considerations [68]. If the change of the mechanical output energy, $\frac{dW_{mech}}{dt}$ is equal to mechanical power, p_{mech} , then:

$$p_{mech} = \frac{dW_{mech}}{dt} = t_e\omega_r \qquad (5.3.4)$$

By expressing W_{mech} as: $W_{mech} = W_{input} - W_{loss} - W_{field}$ *(5.3.5)*

where: W_{input} is the input electrical energy,

W_{loss} represents the stator and rotor losses,

W_{field} is the magnetic energy stored in the field,

expressing each of these energies and working out $\frac{dW_{mech}}{dt}$ then torque is given as:

$$t_e = -\frac{3}{2}pp\bar{\Psi}_r' \times \bar{i}_r' \qquad (5.3.6)$$

where: pp is the number of pole pairs. It can be further developed to obtain a formula similar to *(5.3.2)*:

$t_e = -\frac{3}{2}pp(L_r\bar{i}_r' + L_m\bar{i}_s)\times\bar{i}_r' = -\frac{3}{2}ppL_m\bar{i}_s\times\bar{i}_r'$ since the vector product $\bar{i}_r'\times\bar{i}_r'$ gives zero and further:

$$t_e = -\frac{3}{2}ppL_m\bar{i}_s\times\bar{i}_r' = -\frac{3}{2}pp\frac{L_m}{L_s}(L_s\bar{i}_s + L_m\bar{i}_r')\times\bar{i}_r' = -\frac{3}{2}pp\frac{L_m}{L_s}\bar{\Psi}_s\times\bar{i}_r' \qquad (5.3.7)$$

Since linear magnetic conditions are assumed, then L_m and L_s are constant and the quantity $-\frac{3}{2}\frac{L_m}{L_s}$ is identified with the constant c, thus validating the equation *(5.3.2)*. Therefore torque is given by the interaction of the stator flux with the rotor currents. As the supply is via the stator windings and rotor currents are induced, controlling the torque may be a very tedious task when stator stationary frame is employed. Consequently all equations are transformed onto a general reference frame introduced subsequently .

5.4 The general reference frame (xy)

This section introduces a general reference frame shown in *Figure 5.4.1*. It is referred to by (*xy*) and it rotates at ω_g. Using the general reference frame and expressing all quantities with respect to this frame, different forms of vector control can be then implemented. When for example, the rotor flux oriented control is implemented, the general reference frame is chosen to be fixed to the rotor flux space phasor and consequently all formulas valid in the general reference frame are easily transformed to the required reference frame.

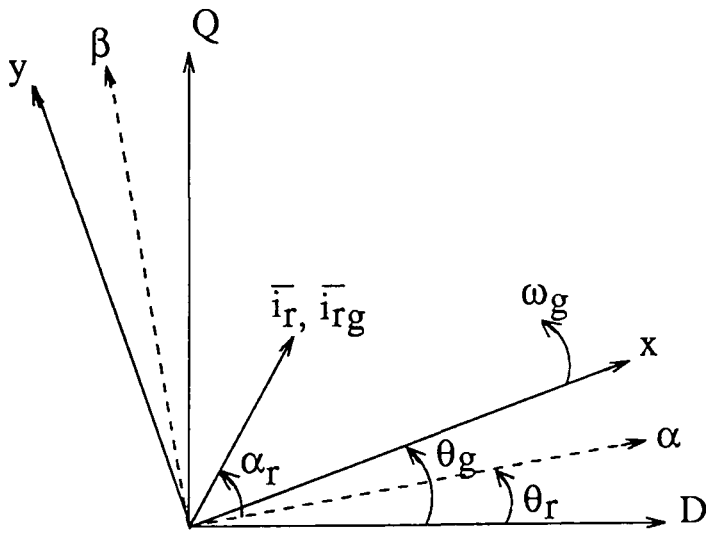

Figure 5.4.1 - The general reference frame

The general reference frame has *xy* as the direct and quadrature axes rotating at ω_g,

$$\omega_g = \frac{d\theta_g}{dt} \qquad (5.4.1)$$

where: θ_g is the angle between the stator *D*-axis and *x*-axis .

The stator quantities are expressed in the general reference frame as:

$$\bar{i}_{sg} = \bar{i}_s e^{-j\theta_g} = i_{sx} + j\, i_{sy} \tag{5.4.2}$$

$$\bar{v}_{sg} = \bar{v}_s e^{-j\theta_g} = v_{sx} + j\, v_{sy} \tag{5.4.3}$$

$$\bar{\Psi}_{sg} = \bar{\Psi}_s e^{-j\theta_g} = \Psi_{sx} + j\Psi_{sy} \tag{5.4.4}$$

and similarly, the rotor quantities:

$$\bar{i}_{rg} = \bar{i}_r e^{-j(\theta_g - \theta_r)} = i_{rx} + j\, i_{ry} \tag{5.4.5}$$

$$\bar{v}_{rg} = \bar{v}_r e^{-j(\theta_g - \theta_r)} = v_{rx} + j\, v_{ry} \tag{5.4.6}$$

$$\bar{\Psi}_{rg} = \bar{\Psi}_r e^{-j(\theta_g - \theta_r)} = \Psi_{rx} + j\Psi_{ry} \tag{5.4.7}$$

Substituting equations *(5.4.2)-(5.4.7)* in *(5.2.1)* and *(5.2.2)* the stator and rotor space phasor voltage equations expressed in the general reference frame are obtained as follows:

$$\bar{v}_{sg} = R_s \bar{i}_{sg} + \frac{d\bar{\Psi}_{sg}}{dt} + j\omega_g \bar{\Psi}_{sg} \tag{5.4.8}$$

$$\bar{v}_{rg} = R_r \bar{i}_{rg} + \frac{d\bar{\Psi}_{rg}}{dt} + j(\omega_g - \omega)\bar{\Psi}_{rg} \tag{5.4.9}$$

The electromagnetic torque in the general reference frame is defined using the equations in *(5.4.5)*, *(5.4.7)* and *(5.3.6)* as follows:

$$t_e = -\frac{3}{2} pp \bar{\Psi}_{rg} \times \bar{i}_{rg} \tag{5.4.10}$$

According to the principle of action-reaction another expression for the torque can be obtained from *(5.4.10)*:

$$t_e = \frac{3}{2} pp \bar{\Psi}_{sg} \times \bar{i}_{sg} \tag{5.4.11}$$

The expression in *(5.4.11)* is further developed:

$$t_e = \frac{3}{2} pp L_m \bar{i}_{rg} \times \bar{i}_{sg} = \frac{3}{2} pp \frac{L_m}{L_r} (L_r \bar{i}_{rg} + L_m \bar{i}_{sg}) \times \bar{i}_{sg} = \tag{5.4.12}$$

$$= \frac{3}{2} pp \frac{L_m}{L_r} \bar{\Psi}_{rg} \times \bar{i}_{sg}$$

(5.4.2) and *(5.4.7)* are substituted in *(5.4.12)* giving *(5.4.13)*.

$$t_e = \frac{3}{2} pp \frac{L_m}{L_r} (\psi_{rx} i_{sy} - \psi_{ry} i_{sx}) \qquad (5.4.13)$$

Thus, torque control can be achieved through the expression *(5.4.13)*. If the reference frame is fixed to the rotor-flux vector, then ψ_{ry} is *0* and torque control can be achieved via ψ_{rx} and i_{sy} components.

5.5 Rotor flux oriented vector control

The general reference frame denoted by (*xy*) axes and rotating at ωg is now considered fixed to the rotor flux-linkage space phasor as shown in *Figure 5.5.1* .

When the reference frame is fixed to the rotor flux space phasor then it rotates at ω_{mr}, defined as:

$$\omega_{mr} = \frac{d\rho_r}{dt} \qquad (5.5.1)$$

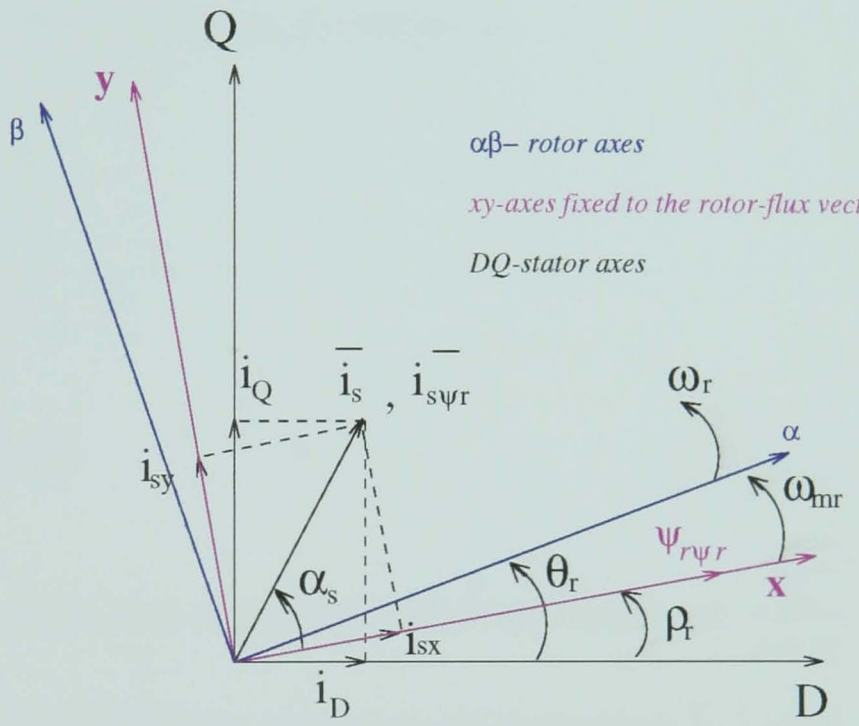

Figure 5.5.1 - The reference frame fixed to the rotor flux- linkage space phasor

The torque is defined from *(5.4.13)* as:

$$t_e = \frac{3}{2} pp \frac{L_m}{L_r} \psi_{rx} i_{sy} \qquad (5.5.2)$$

since ψ_{ry}=0, meaning that the rotor linkage flux space phasor has only a *x*-axis component. The rotor linkage flux space phasor is given by:

$$\bar{\psi}_{r\psi r} = \psi_{rx} = |\bar{\psi}_r| = L_r \bar{i}_{r\psi r} + L_m \bar{i}_{s\psi r} \qquad (5.5.3)$$

The rotor magnetising current is defined as:

$$\bar{i}_{mr} = \frac{\bar{\psi}_{r\psi r}}{L_m} = \frac{L_r}{L_m} \bar{i}_{r\psi r} + \bar{i}_{s\psi r} = \bar{i}_{s\psi r} + (1 + \sigma_r) \bar{i}_{r\psi r} \qquad (5.5.4)$$

where: σ_r is the rotor leakage factor defined by:

$$\sigma_r = \frac{L_{rl}}{L_m} \qquad (5.5.5)$$

where L_{rl} is the rotor leakage inductance.

From *(5.5.4)*:

$$\bar{\Psi}_{r\psi r} = L_m \bar{i}_{mr} \qquad (5.5.6)$$

and modulus of the rotor flux space phasor is given by:

$$\left|\bar{\Psi}_{r\psi r}\right| = \Psi_{rx} = L_m \left|\bar{i}_{mr}\right| \qquad (5.5.7)$$

Substituting *(5.5.7)* in *(5.5.2)* gives the torque as:

$$t_e = \frac{3}{2} pp \frac{L^2{}_m}{L_r} \left|\bar{i}_{mr}\right| i_{sy} \qquad (5.5.8)$$

or in terms of σ_r:

$$t_e = \frac{3}{2} pp \frac{L_m}{1+\sigma_r} \left|\bar{i}_{mr}\right| i_{sy} \qquad (5.5.9)$$

Equation *(5.5.9)* shows that if magnetic conditions are linear, $\frac{L_m}{1+\sigma_r}$ is a constant quantity and torque can be controlled by independently controlling $\left|\bar{i}_{mr}\right|$ and i_{sy}.

5.5.1 The decoupling circuit

Using the stator voltage equation for the general reference frame *(5.4.8)* , a similar equation can be written for the reference frame fixed to the rotor flux space phasor :

$$\bar{v}_{s\psi r} = R_s \bar{i}_{s\psi r} + L_s \frac{d\bar{i}_{s\psi r}}{dt} + L_m \frac{d\bar{i}_{r\psi r}}{dt} + j\omega_{mr} L_s \bar{i}_{s\psi r} + j\omega_{mr} L_m \bar{i}_{r\psi r} \qquad (5.5.10)$$

where subscript (Ψ_r) means that the quantity is expressed in the rotor flux frame. In a squirrel-cage induction motor the rotor currents cannot be directly measured, so they are

eliminated from the stator voltage equation. From *(5.5.4)* the rotor current, $\bar{i}_{r\psi r}$ can be expressed as:

$$\bar{i}_{r\psi r} = \frac{\bar{i}_{mr} - \bar{i}_{s\psi r}}{1+\sigma_r} \qquad (5.5.11)$$

and is then substituted in *(5.5.10)*. Further, both sides of the equation are divided by the value of the stator resistance R_s yielding:

$$\frac{\bar{v}_{s\psi r}}{R_s} = \left(L_s - \frac{L^2{}_m}{L_r}\right)\frac{1}{R_s}\frac{d\bar{i}_{s\psi r}}{dt} + j\omega_{mr}\left(L_s - \frac{L^2{}_m}{L_r}\right)\frac{1}{R_s}\bar{i}_{s\psi r} + \bar{i}_{s\psi r} + \left(j\omega_{mr}\left|\bar{i}_{mr}\right| + \frac{d\left|\bar{i}_{mr}\right|}{dt}\right)\frac{L^2{}_m}{L_r R_s} \qquad (5.5.12)$$

where: $L_s - \dfrac{L^2{}_m}{L_r} = L_s'$ is the transient stator inductance,

$T_s = \dfrac{L_s}{R_s}$ is the stator time constant,

$T_s' = \dfrac{L_s'}{R_s}$ is the stator transient time constant.

Using these notations *(5.5.12)* becomes:

$$\frac{\bar{v}_{s\psi r}}{R_s} = T_s'\frac{d\bar{i}_{s\psi r}}{dt} + j\omega_{mr}T_s'\bar{i}_{s\psi r} + \bar{i}_{s\psi r} + \left(j\omega_{mr}\left|\bar{i}_{mr}\right| + \frac{d\left|\bar{i}_{mr}\right|}{dt}\right)(T_s - T_s') \qquad (5.5.13)$$

and with some rearrangements give:

$$T_s'\frac{d\bar{i}_{s\psi r}}{dt} + \bar{i}_{s\psi r} = \frac{\bar{v}_{s\psi r}}{R_s} - j\omega_{mr}T_s'\bar{i}_{s\psi r} - (T_s - T_s')\left(j\omega_{mr}\left|\bar{i}_{mr}\right| + \frac{d\left|\bar{i}_{mr}\right|}{dt}\right) \qquad (5.5.14)$$

Equation *(5.5.14)* can now be resolved into its x and y components using the following relationships for the stator current space phasor and stator voltage space phasor:

$$\bar{i}_{s\psi r} = i_{sx} + ji_{sy} \qquad (5.5.15)$$

$$\bar{v}_{s\psi r} = v_{sx} + jv_{sy} \qquad (5.5.16)$$

Taking the real and imaginary components of the resultant equation, two equations are obtained i.e.:

$$T_s' \frac{di_{sx}}{dt} + i_{sx} = \frac{v_{sx}}{R_s} + \omega_{mr} T_s' i_{sy} - (T_s - T_s') \frac{d|\bar{i}_{mr}|}{dt} \qquad (5.5.17)$$

$$T_s' \frac{di_{sy}}{dt} + i_{sy} = \frac{v_{sy}}{R_s} - \omega_{mr} T_s' i_{sx} - (T_s - T_s') \omega_{mr} |\bar{i}_{mr}| \qquad (5.5.18)$$

It can be seen that there is coupling between the stator equations, so that i_{sx}, the flux producing component and i_{sy}, the torque producing component cannot be easily and independently controlled.

If $|\bar{i}_{mr}|$ = constant then $\frac{d|\bar{i}_{mr}|}{dt}$=0 and equations *(5.5.17)* and *(5.5.18)* reduce to:

$$L_s' \frac{di_{sx}}{dt} + R_s i_{sx} = v_{sx} + \omega_{mr} L_s' i_{sy} \qquad (5.5.19)$$

and

$$L_s' \frac{di_{sy}}{dt} + R_s i_{sy} = v_{sy} - \omega_{mr} L_s' i_{sx} - (L_s - L_s') \omega_{mr} |\bar{i}_{mr}| \qquad (5.5.20)$$

which represents the motor operating under constant rotor flux conditions.

From *(5.5.19)* and *(5.5.20)* it is seen that the rotational voltage components are:

$$v_{rot\_x} = -\omega_{mr} L_s' i_{sy} \qquad (5.5.21)$$

$$v_{rot\_y} = \omega_{mr} L_s' i_{sx} + (L_s - L_s') \omega_{mr} |\bar{i}_{mr}| \qquad (5.5.22)$$

The outputs of the current controllers, denoted by v_{cc_x} and v_{cc_y} are given by:

$$v_{cc\_x} = L_s' \frac{di_{sx}}{dt} + R_s i_{sx} \qquad (5.5.23)$$

$$v_{cc\_y} = L_s' \frac{di_{sy}}{dt} + R_s i_{sy} \qquad (5.5.24)$$

Thus the stator voltage components are calculated as follows:

$$v_{sx} = v_{cc\_x} + v_{rot\_x} \tag{5.5.25}$$

$$v_{sy} = v_{cc\_y} + v_{rot\_y} \tag{5.5.26}$$

In *Figure 5.5.2* a Simulink block diagram is presented from which v_{rot_x} and v_{rot_y} can be calculated provided that i_{sx} , i_{sy}, ω_{mr} and $|\bar{i}_{mr}|$ are known.

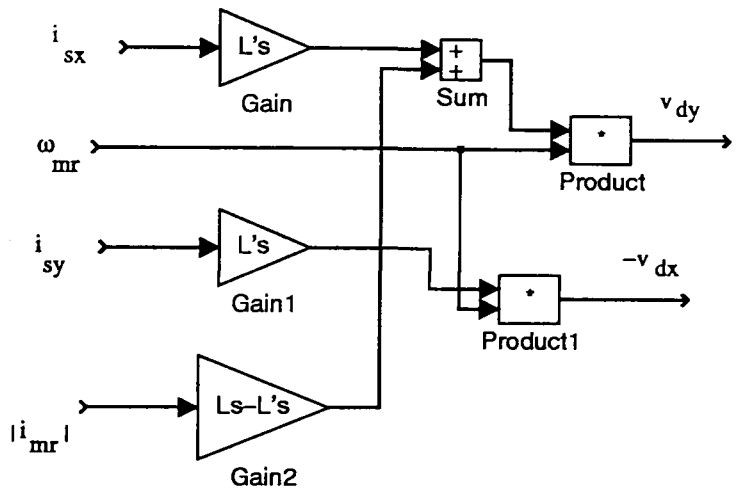

Figure 5.5.2 - A Simulink scheme to calculate the decoupling voltages

The currents on (xy) axes can be calculated from the (DQ) currents by employing the following relations:

$$i_{sx} = i_D \cos \rho_r + i_Q \sin \rho_r \tag{5.5.27}$$

$$i_{sy} = -i_D \sin \rho_r + i_Q \cos \rho_r \tag{5.5.28}$$

However ω_{mr} and $|\bar{i}_{mr}|$ are more troublesome to be estimated and they may be found out through a flux model. This is given in the subsequent section.

5.5.2 The flux model

The modulus and phase angle of the rotor flux space phasor are needed to obtain ω_{mr} and $|\bar{i}_{mr}|$. A flux model which is used to calculate $|\bar{\psi}_{r\psi r}|$ and ρ_r is presented in this section. For this purpose the rotor voltage equation is used. Using the rotor voltage equation in the

general reference frame showed in *(5.4.9)*, a similar equation in the reference frame fixed to the rotor flux space phasor is obtained:

$$0 = R_r \bar{i}_{r\psi r} + \frac{d\bar{\psi}_{r\psi r}}{dt} + j(\omega_{mr} - \omega_r)\bar{\psi}_{r\psi r} \qquad (5.5.29)$$

By substituting the rotor flux expression *(5.5.7)* in *(5.5.29)* it gives:

$$0 = R_r \bar{i}_{r\psi r} + L_m \frac{d|\bar{i}_{mr}|}{dt} + j(\omega_{mr} - \omega_r) L_m |\bar{i}_{mr}| \qquad (5.5.30)$$

and further $\bar{i}_{r\psi r}$ is substituted using *(5.5.10)* :

$$0 = R_r \frac{L_m}{L_r}\left(|\bar{i}_{mr}| - \bar{i}_{s\psi r}\right) + L_m \frac{d|\bar{i}_{mr}|}{dt} + j(\omega_{mr} - \omega_r) L_m |\bar{i}_{mr}| \qquad (5.5.31)$$

Multiplying both sides of the equation in *(5.5.31)* by the quantity $\frac{L_r}{L_m R_r}$ and rearranging is obtained:

$$T_r \frac{d|\bar{i}_{mr}|}{dt} + |\bar{i}_{mr}| = \bar{i}_{s\psi r} - j(\omega_{mr} - \omega_r) T_r |\bar{i}_{mr}| \qquad (5.5.32)$$

where T_r denotes the rotor time constant, given by:

$$T_r = \frac{L_r}{R_r} \qquad (5.5.33)$$

Resolving *(5.5.32)* into its x and y components the following equations are obtained:

$$T_r \frac{d|\bar{i}_{mr}|}{dt} + |\bar{i}_{mr}| = i_{sx} \qquad (5.5.34)$$

$$\omega_{mr} = \omega_r + \frac{i_{sy}}{T_r |\bar{i}_{mr}|} \qquad (5.5.35)$$

Based on these two equations, a rotor flux model can be designed in order to calculate $|\bar{\psi}_{r\psi r}|$, the modulus of the rotor flux space phasor and ρ_r, the phase angle of $\bar{\psi}_{r\psi r}$.

In *Figure 5.5.3* a Simulink flux model is proposed. Part of the scheme in *Figure 5.5.3* , there is the block which realises the transformation between quantities on *DQ* axis and quantities on *xy* axis. This block is used here to transform the currents, according to the relations in *(5.5.27)* and *(5.5.28)* and its expanded form is given in *Figure 5.5.4.*

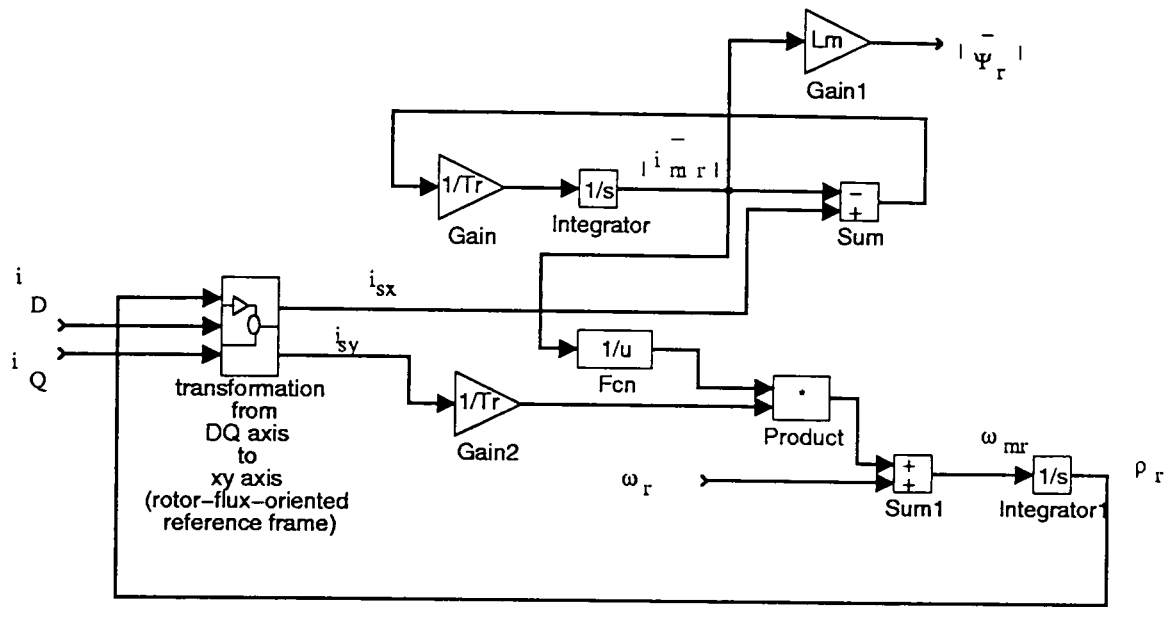

Figure 5.5.3 - A Simulink block diagram for the flux model

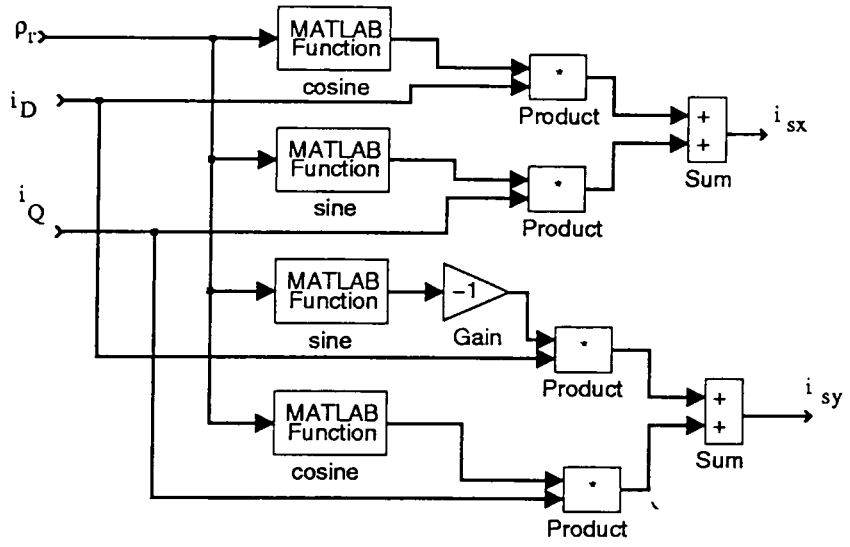

Figure 5.5.4 - The internal structure of the`Decoupling block`

Other flux models can be designed depending on which inputs are chosen or available. A scheme for the rotor flux oriented control of an induction motor can be patched up by combining the decoupling scheme and the rotor flux model. In this scheme, shown in *Figure 5.5.5* , there is also a function generator used for field-weakening purposes. Above a certain frequency, an increase in speed is only possible by decreasing the rotor flux, i.e. by decreasing $|i_{mr}|$. Thus by using the function generator, based on the value of the rotor speed, the reference value for $|i_{mr}|$ may be calculated.

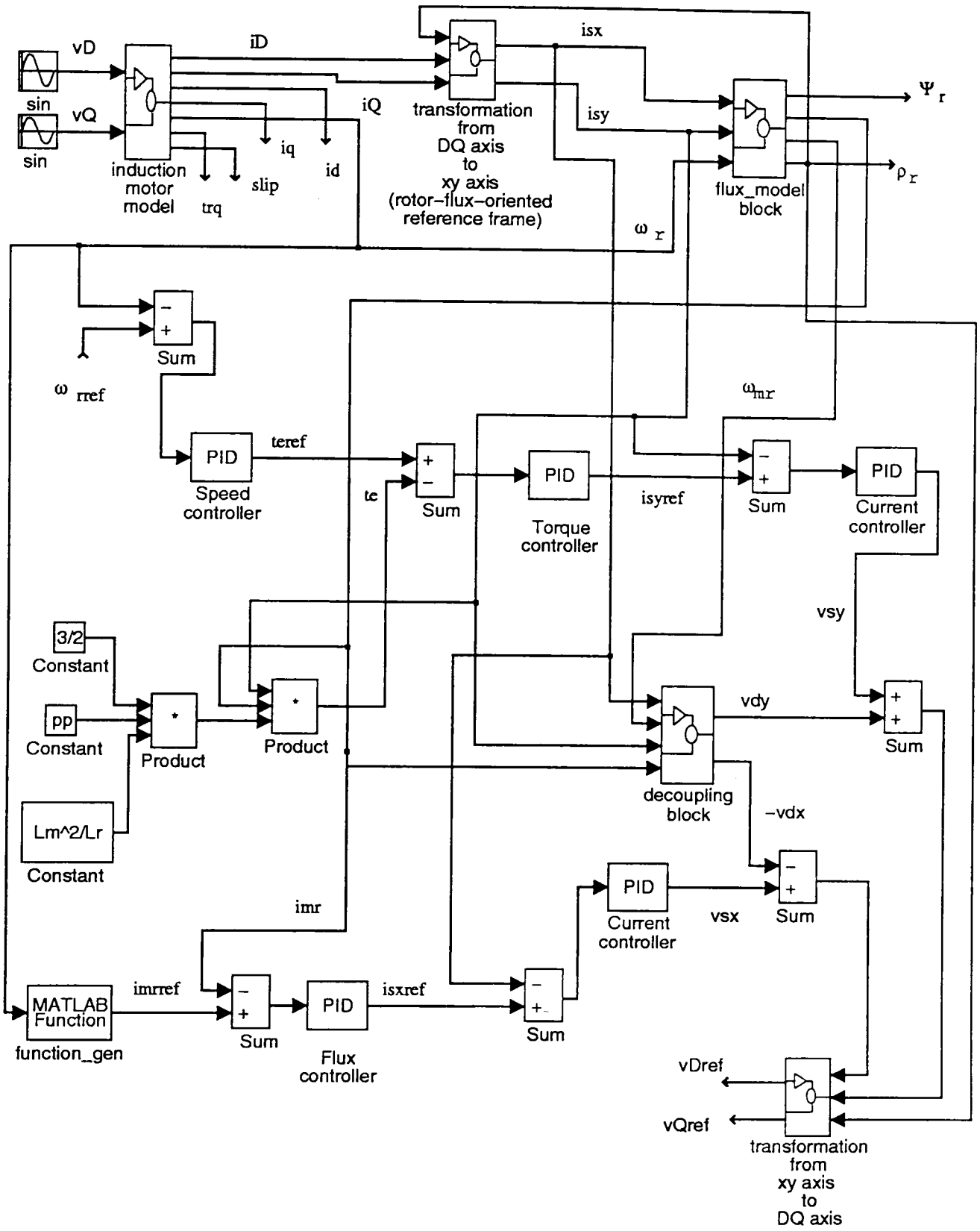

Figure 5.5.5 - A rotor flux oriented vector control scheme

In this chapter the principle of rotor flux oriented control has been introduced. Voltage equations are given in a general reference frame which is subsequently aligned to the rotor flux vector. A flux model is needed in order to achieve indirect vector control. This flux model requires information about the motor parameters in order to estimate the rotor flux and its phase angle. Motor parameter estimation is reported and some procedures, developed and tested by the author, are introduced.

Chapter 6

Motor Parameter Estimation

6.1 Introduction

Full advantage of field orientation control is obtained only if it is possible to know accurately the instantaneous position of the rotor flux vector relative to a stationary reference frame. The rotor flux estimation can be carried out in two different ways: direct measurement by using sensors, estimators or both and indirect measurement which combines slip calculation with a rotor position or speed measurement. Slip calculation relies upon rotor time constant information, therefore there is considerable interest in estimating this parameter on-line.

Full use of the advantages of modern a.c. drives with field-oriented control can only be made if during commissioning, the control system is accurately adapted to the connected motor.

In many applications, when the motor and the inverter are not sold together as a unit or when the motor has to be replaced, the parameters of the motor are not known beforehand.

It then requires intensive testing of the machine and controller tuning, process that can be expensive, time consuming and it may call for specially trained staff. Thus self-commissioning systems have been developed, in which the system itself determines the machine parameters during the commissioning period and accordingly sets the control parameters needed in the control scheme.

The concept adopted for the estimation measurements is to use the same inverter to perform some tests and derive essential parameter values before actually starting the drive function. Single phase excitation is normally employed to obtain locked rotor tests with the tests requiring a series of programmed voltage and current commands to be delivered to the inverter. Thus using the inverter of the drive itself has the advantage of being a cheaper solution and also providing a better operation, since the parts constituting the drive are the same either when doing the tests or performing a mechanical task.

In this chapter the bases of parameter estimation are given in a paper by H. Schierling [37] The approach outlined in this paper together with information supplied in [68] has been used as the basis for carrying out simulation tests and experimental implementations for estimation of the motor electrical parameters .

6.2 Parameter Estimation - Basics

In the case of an a.c. motor, the electrical parameters are generally determined using the classical locked rotor and no-load tests. From the locked rotor test the rotor resistance can be calculated provided that a *d.c.* test has been done to measure the stator resistance, and the total leakage reactance. Further, knowing the *Nema* design letter of the machine, the total reactance could be split thereby allowing rotor and stator reactance estimation. The no-load test permits the magnetising reactance to be found. However the classical tests are difficult to be performed and in some cases , the parameters obtained have to be modified to consider for example, temperature changes during running.

The tests discussed further in this chapter, realised by using the same inverter that supplies the *a.c.* motor , are aiming at estimating the total leakage inductance, rotor and stator

resistances and the rotor time constant. The relations defining the motor operation and the practical schemes necessary to perform the tests are presented in the following subsections.

6.2.1 Mathematical relations of induction motor

The following system of equations describes the squirrel-cage motor dynamics in the stator reference frame [37]:

$$\begin{bmatrix} L_\sigma \cdot \frac{di_\alpha}{dt} \\ L_\sigma \cdot \frac{di_\beta}{dt} \\ \frac{d\psi_\alpha}{dt} \\ \frac{d\psi_\beta}{dt} \end{bmatrix} = \begin{bmatrix} -R_S - R_R & 0 & 1/T_R & \omega_m \\ 0 & -R_S - R_R & -\omega_m & 1/T_R \\ R_R & 0 & -1/T_R & -\omega_m \\ 0 & R_R & \omega_m & -1/T_R \end{bmatrix} \cdot \begin{bmatrix} i_\alpha \\ i_\beta \\ \psi_\alpha \\ \psi_\beta \end{bmatrix} + \begin{bmatrix} u_\alpha \\ u_\beta \\ 0 \\ 0 \end{bmatrix} \qquad (6.2.1)$$

where: R_S, R_R are the stator resistance and the transformed rotor resistance respectively; the rotor resistance is transformed by quotient $K_R = \frac{L_M}{L_R}$ squared;

u_α, u_β are the $\alpha - \beta$ components of stator voltage ;

i_α, i_β are the $\alpha - \beta$ components of stator current ;

ψ_α, ψ_β are the $\alpha - \beta$ components of transformed rotor flux ; transformation done by K_R;

ω_m is the mechanical angular frequency transformed for one pole pair ;

$T_R = \frac{L_R}{R_R}$ is the rotor time constant ;

L_R is the transformed rotor inductance, transformation done by K_R squared ;

L_σ is the total leakage inductance.

As the tests are carried out at standstill, that is $\omega_m = 0$, the equations for the α -axis can be rewritten as follows:

$$\begin{bmatrix} L_\sigma \cdot \dfrac{di_\alpha}{dt} \\ \dfrac{d\psi_\alpha}{dt} \end{bmatrix} = \begin{bmatrix} -R_S - R_R & 1/T_R \\ R_R & -1/T_R \end{bmatrix} \cdot \begin{bmatrix} i_\alpha \\ \psi_\alpha \end{bmatrix} + \begin{bmatrix} u_\alpha \\ 0 \end{bmatrix} \qquad (6.2.2)$$

The equation system of *(6.2.2)* allows under certain conditions the estimation of some motor parameters. The theoretical relations and procedures utilised for these estimations are presented in the subsections to follow.

6.2.2 Measurement of the total leakage inductance

For a time interval shorter than T_R the motor behaves like a first-order delay element and therefore the initial rise of the current is determined by L_σ. Thus the following relation can be deducted from *(6.2.2)* :

$$L_\sigma = \frac{u_\alpha}{\dfrac{di_\alpha}{dt}} \qquad (6.2.3)$$

The measuring procedure consists of applying certain voltage pulses at certain time instants. At a moment, t_1 the voltage $u_\alpha = (2/3) \cdot u_{dc}$ is applied to the motor (where u_{dc} is the *d.c.* link voltage) and the current starts to rise. When the stator current reaches the peak value of the rated current, at moment t_2, then $u_\alpha = 0$ is applied and the stator terminals are short-circuited. At the moment in time denoted by t_3, $u_\alpha = -(2/3) \cdot u_{dc}$ is applied and the stator current will continue decreasing until reaches the negative of the peak value. At that instant, t_4, the motor is short circuited and remains so from that point on.

Figure 6.2.1 shows the way in which the voltage pulses are applied to the induction motor during the test sequence.

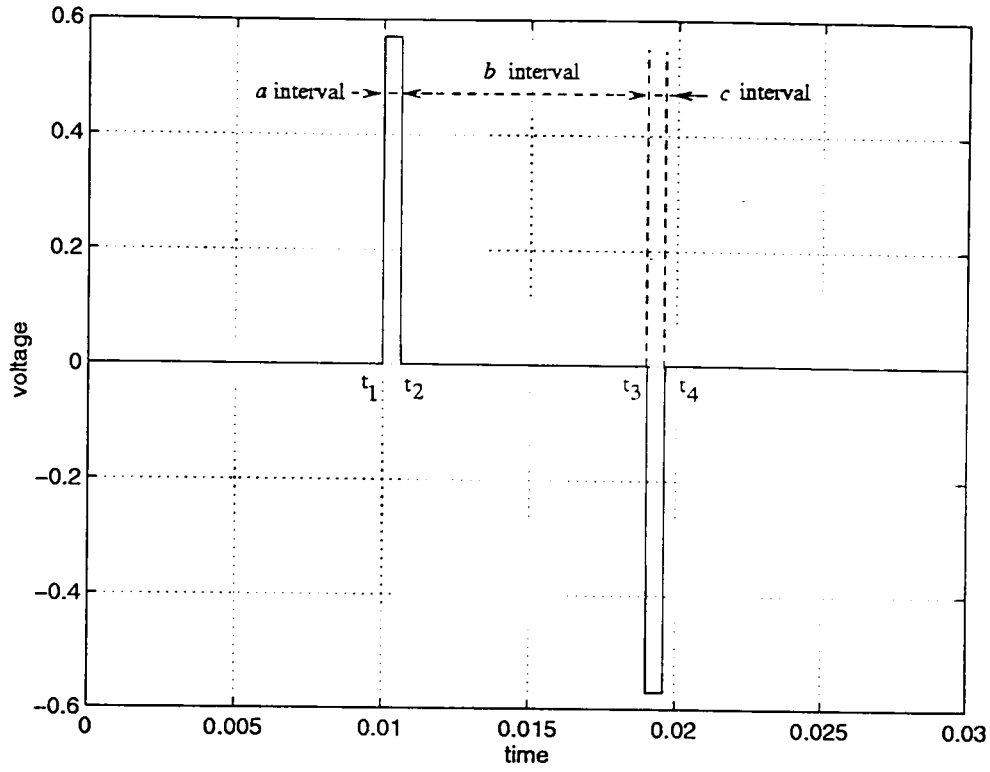

Figure 6.2.1 - The voltage pulses applied to the induction motor

The current response of the motor is shown in *Figure 6.2.2*.

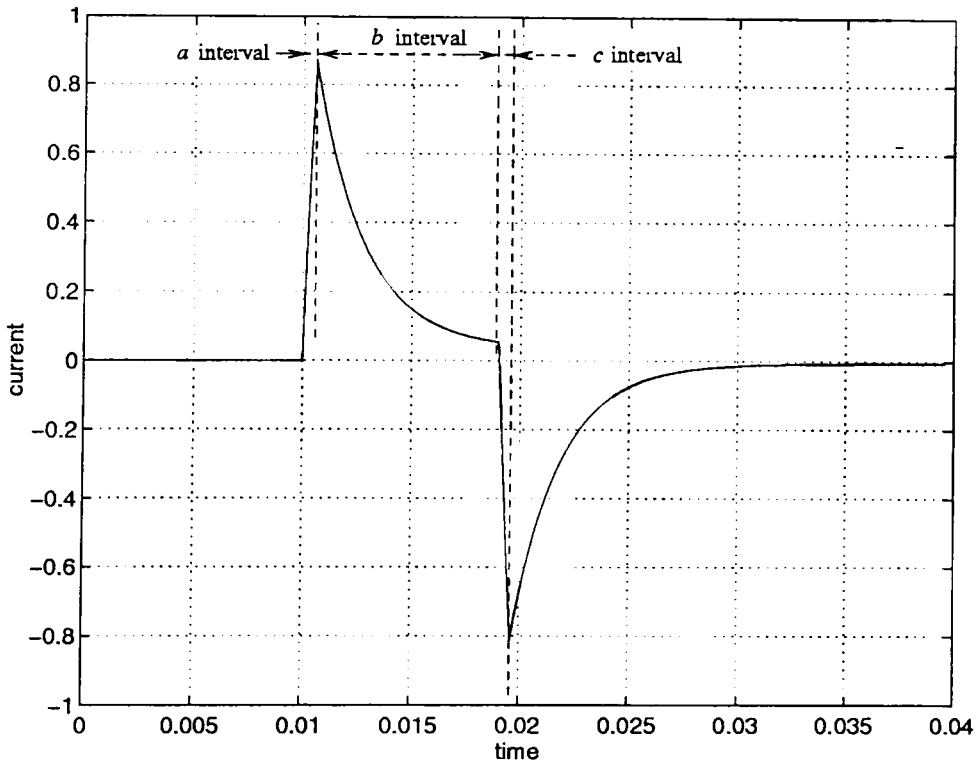

Figure 6.2.2 - The current response in the induction motor

The peak value of current is not necessarily reached during these tests, as the voltage is switched to *0* before the current reaches *1 p.u.* This occurs because the current is rapidly increasing and with a sampling time of *426.66* μs the value of *1 p.u.* can not be reached. That means that if at the end of a certain sampling period the current has reached *0.8 p.u.* and the simulation and also practical test was carried out for one more sampling step, the

current would reach a exceedingly high value, possibly around *1.5 p.u.* or *2 p.u.*. However using a sampling time of *200* μs, values close to *1 p.u.* can be obtained. Obviously, a smaller sampling step would be more appropriate, yet this is not possible because of the algorithm length (in microprocessor clock cycles). In *Figure 6.2.2* the current is shown to have values around *0.8 p.u.*.

Using the equation *(6.2.3)* , the total leakage inductance can be expressed by:

$$L_\sigma = \frac{2}{3} u_{dc} \frac{t_4 - t_3}{i_\alpha(t_3) - i_\alpha(t_4)} \qquad (6.2.4)$$

This equation has been derived assuming ideal conditions, neglecting the effects of magnetic saturation and eddy currents.

The scheme used for implementing the test procedure is presented in *Figure 6.2.3*. It has phase current measurements to determine $i_\alpha(t_3)$ and $i_\alpha(t_4)$ and a PWM modulation block which sends the appropriate firing signals to the inverter.

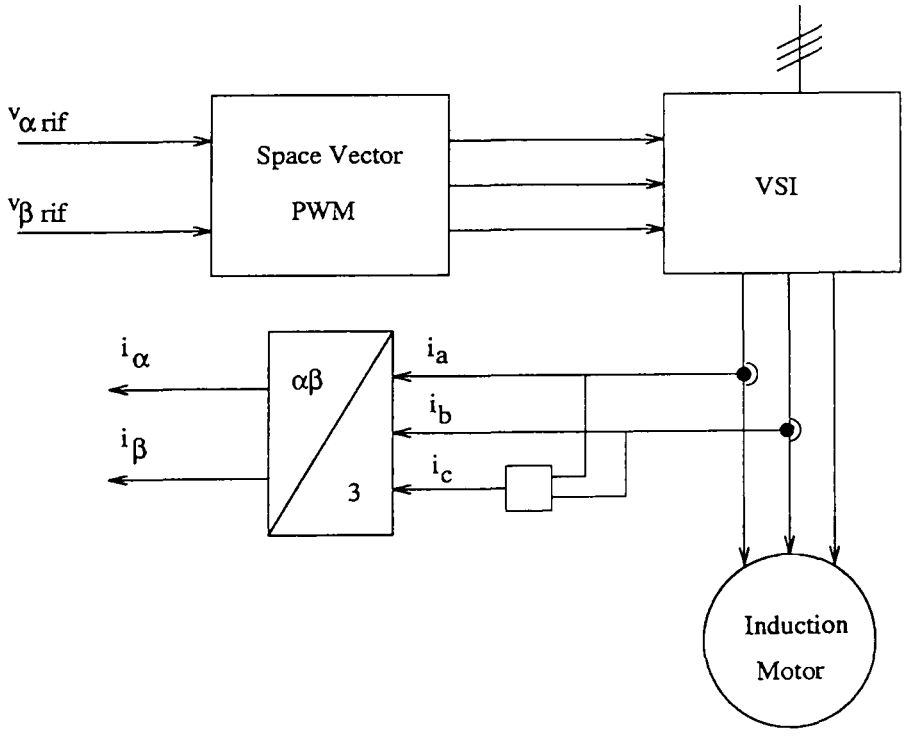

Figure 6.2.3 - The scheme used for implementing the total leakage reactance estimation procedure

This is actually a part of the vector control scheme simulated in *Pascal* language and implemented using assembler language on a Siemens *SAB 80C166* microcontroller. The total scheme can be seen in *Figure 6.2.4*.

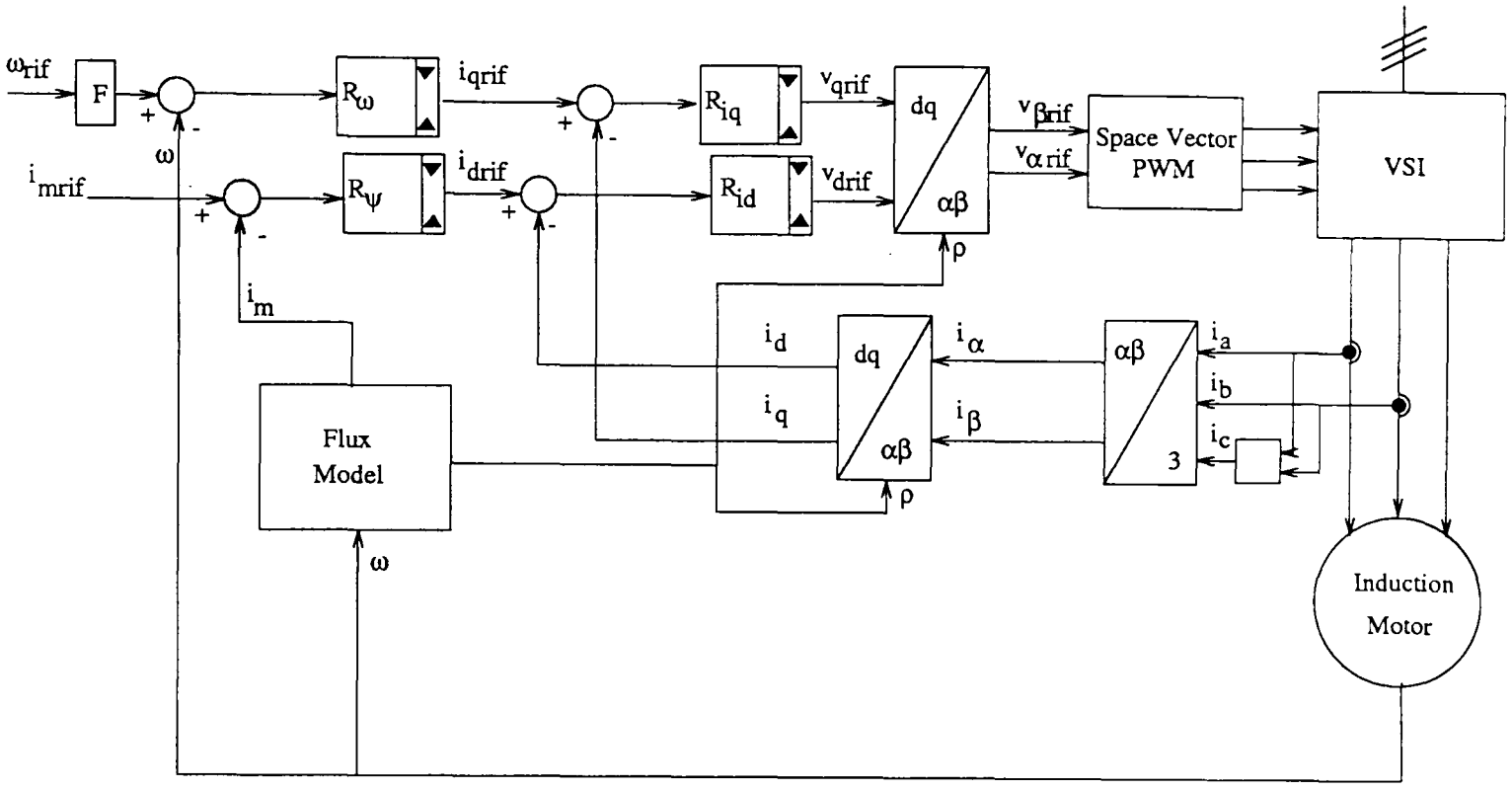

Figure 6.2.4 - The vector control scheme

Using the scheme of *Figure 6.2.3*, the estimating procedure for L_σ was first simulated in *Pascal* and then implemented for the *SAB* microcontroller. The assembler program listing can be found in Appendix 3. Simulation results are presented in *Section 6.3* and experimental results in *Section 6.4*.

6.2.3 Measurement of resistances and rotor time constant

Estimation of the stator and rotor resistances and the rotor time constant can be obtained by impressing different values of *d.c.* current on the motor. This can be performed by three current pulses of different values used as references which are sent to the i_D and i_Q current controllers. The reference voltages constituting the output of the current controllers are then sent to the PWM modulator and further to the motor. The sequence of commands is shown in *Figure 6.2.5* and the reference voltages actually applied to the motor can be seen in *Figure 6.2.6*.

In the first interval (*a*) the current imposed will take values between *0.15 p.u.* and *0.3 p.u.*. This value is chosen so that it should be less than the magnetising current of the machine to assure that the effects of the magnetic saturation will not affect parameter estimation. As

the magnetising current would not be known, the *d.c.* current applied may be taken as a third of the peak of the rated current of the machine.

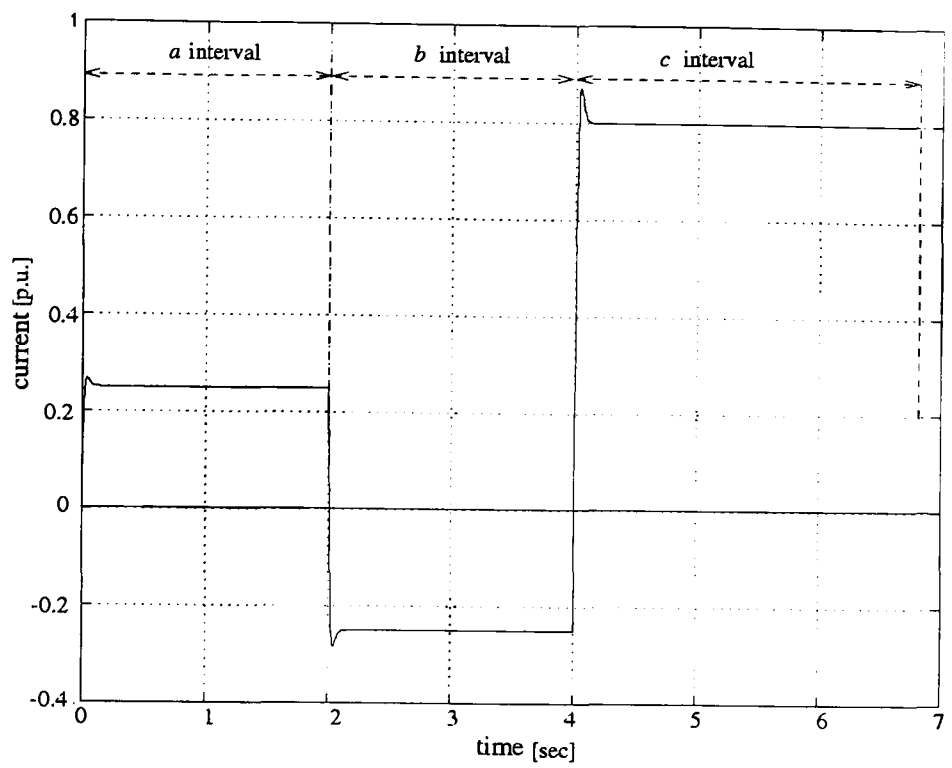

Figure 6.2.5 - The current pulses

After steady-state has been reached a current pulse having the negative value of the *(a)* interval current is applied. This constitutes the interval *(b)*. The current of the last interval, (*c*), is between *0.8 p.u* and *1 p.u.*.

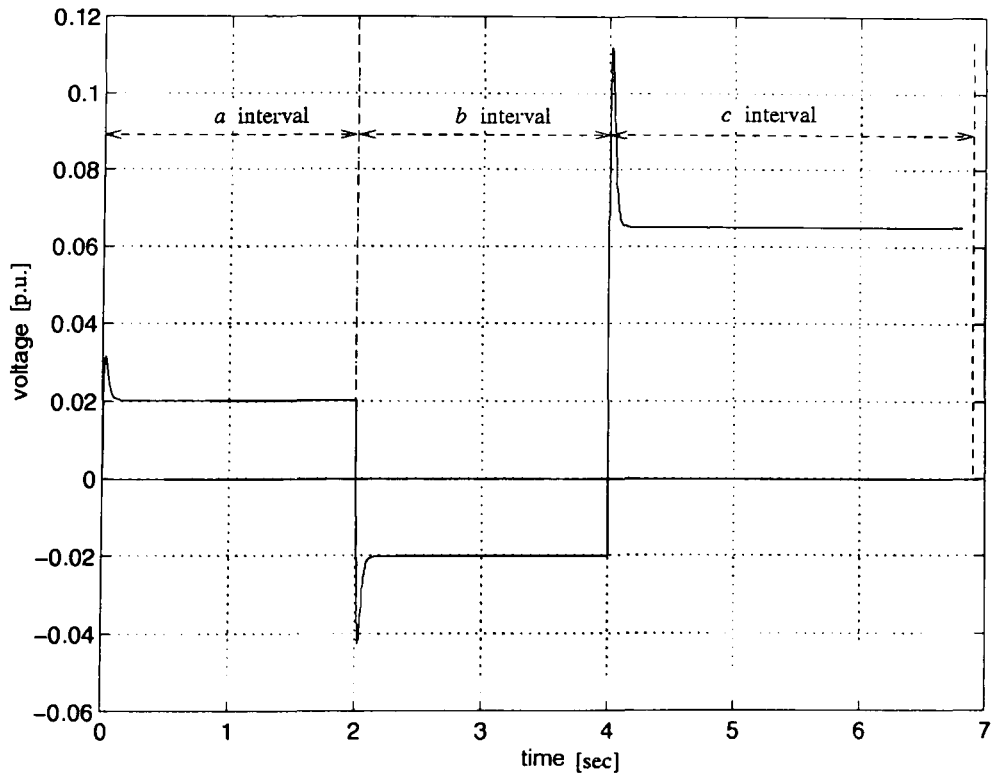

Figure 6.2.6 - The voltage response in the induction motor

The test results of this procedure are used for estimating all three parameters. Initially, using test results from all three time intervals, the stator resistance is estimated. Following this, the rotor resistance is calculated, using the previously calculated stator resistance and results from the beginning of the *(c)* interval. Finally, data from the *(b)* interval are considered and the rotor time constant is evaluated.

To implement the test sequence presented so far, a scheme similar to the one in *Figure 6.2.3* is employed. In addition to *Figure 6.2.3*, current controllers are connected as this test is based on reference current pulses.

The scheme is reported in *Figure 6.2.7*, where $i_{\alpha rif}$ and $i_{\beta rif}$ are the *d.c.* current references and $i_{\alpha rif}$ will take values from *0.05 p.u.* to *0.3 p.u.* and $i_{\beta rif}$ will be *0*.

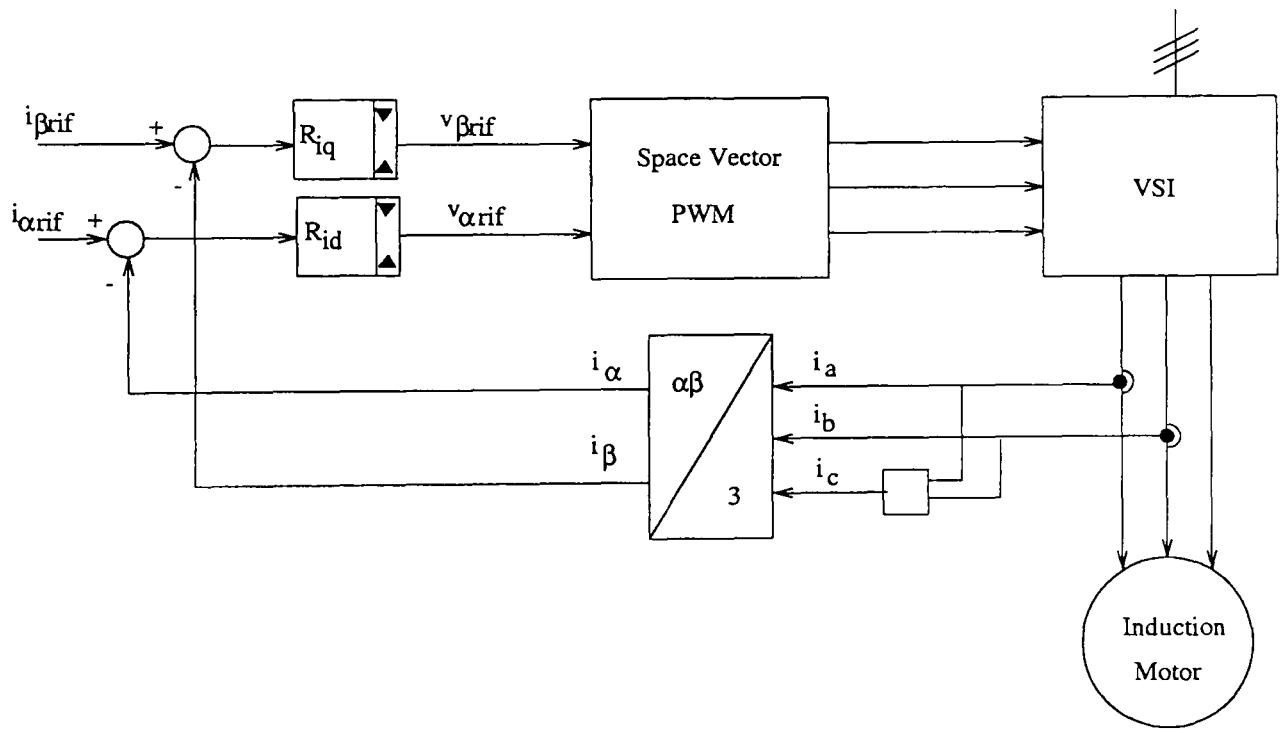

Figure 6.2.7 - The scheme used for implementing the resistances and rotor time constant estimation procedures

Having discussed the general test procedure and the current pulse sequence applied to the motor through the current controllers and PWM modulator, details on parameter calculation are now given.

- **Estimation of the stator resistance :**

When the steady state is reached, i_α and ψ_α can be considered constant and consequently the equations of *(6.2.2)* are reduced to:

$$u_\alpha = R_S \cdot i_\alpha \qquad (6.2.5)$$

Thus the stator resistance is given by the relationship:

$$R_S = \frac{u_{\alpha(c)} - u_{\alpha(a)}}{i_{\alpha(c)} - i_{\alpha(a)}} \qquad (6.2.6)$$

where subscripts (a) refer to *(a)* interval quantities and (c) refer to *(c)* interval quantities.

- **Estimation of the rotor resistance:**

The formula used to calculate the rotor resistance is derived by rearranging the expression for the voltage at the beginning of the *(c)* interval :

$$u_{\alpha(c)} = R_S \cdot i_{\alpha(c)} + R_R(i_{\alpha(c)} - i_{\alpha(b)}) \qquad (6.2.7)$$

and consequently:

$$R_R = \frac{u_{\alpha(c)} - R_S \cdot i_{\alpha(c)}}{(i_{\alpha(c)} - i_{\alpha(b)})} \qquad (6.2.8)$$

- **Estimation of the rotor time constant :**

The rotor flux changes exponentially from its initial value in interval *(a)* to its steady state value in interval *(b)* and this transition is mirrored by the terminal voltage. The voltage response at the beginning of the *(b)* interval is shown in *Figure 6.2.8.* The time constant of this exponential function is the rotor time constant.

The rotor time constant evaluation is graphically explained in *Figure 6.2.8.* The exponential function is analysed between the time moment called *timp_minUsaRef*, when the minimum voltage of (*b*) interval occurs (*MinUsaRef*) and the time when (*b*) interval voltage reaches the steady state value: *MaxUsaRef* .

Generally, if an exponential curve is defined by:

$$y(t) = y_{\max}(1 - e^{-\frac{t}{\tau}}) \qquad (6.2.9)$$

Then the time constant, τ can be worked out graphically by calculating $y(t)$ when t= τ. Thus, $y(\tau) = y_{\max}(1 - \frac{1}{e})$ on the ordinate axis gives τ on the abscise (time-axis).

In the case of the rotor flux curve, the y corresponding to the time constant is noted *UsaRef67* and is calculated by:

$$UsaRef67 = MaxUsaRef + (MinUsaRef - MaxUsaRef) \cdot \frac{1}{e} \qquad (6.2.10)$$

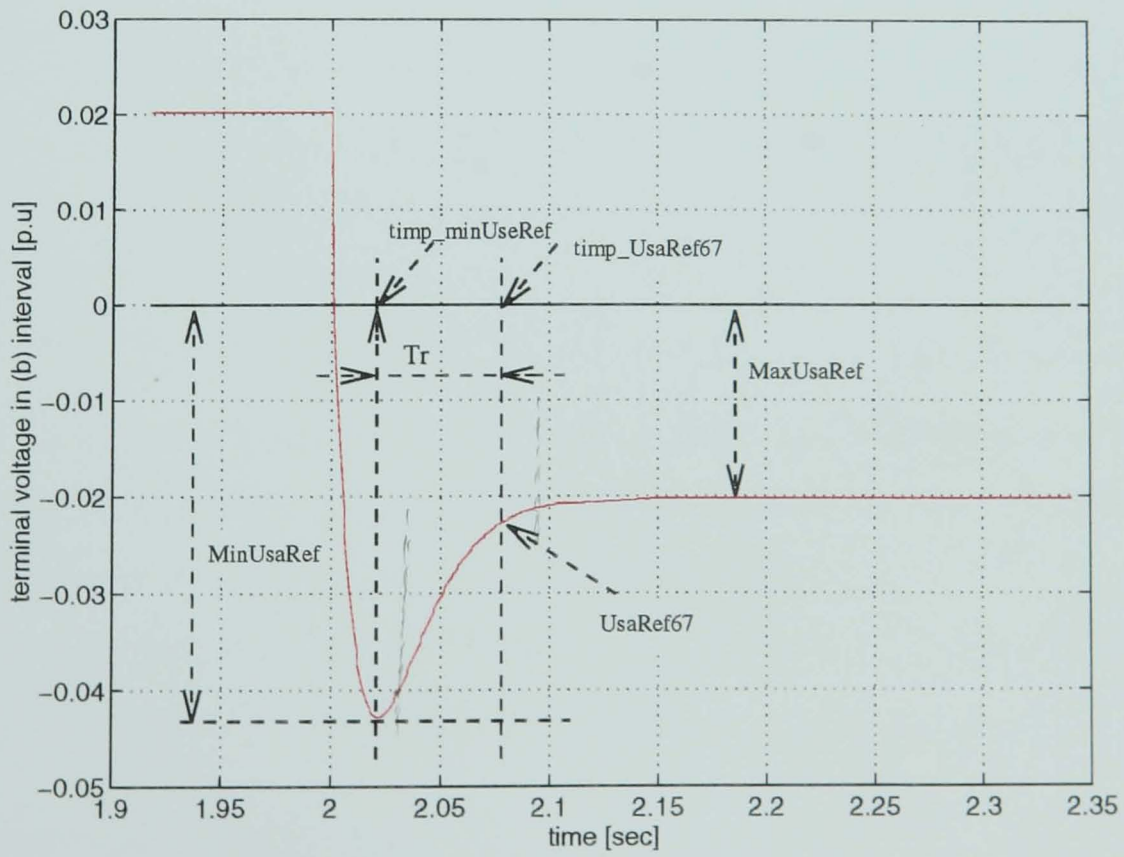

Figure 6.2.8 - The voltage response in (b) interval and the graphical evaluation of the rotor time constant

The time when *UsaRef67* occurs is *timp_UsaRef67*, and T_R, the rotor time constant is given:

$$T_R = timp\_UsaRef67 - timp\_minUsaRef \qquad (6.2.11)$$

The simulation and experimental results obtained are reported in the *Section 6.3* and *Section 6.4*.

6.3 Simulation Results

Section 6.2 has provided the theoretical basis and motor equations needed for the motor parameter estimation proposed. The present section presents simulation results obtained when the estimating procedures were implemented and tested on different motors. This work has been carried out at the Department of Electrical Engineering within the *University of L'Aquila, Italy*. As their research group had already developed a simulation program written in Pascal language to model an inverter-fed motor, these procedures were also implemented using the same programming language. They were designed to work together with the available simulation program.

To enable a good comparison, simulation results for three different motors are given in this section. Next *Section 6.4* presents practical results of a real-time implementation using the motor no. 1 [Appendix 4].

6.3.1 Simulation Results - part I -

Simulations were carried out using the data set of the motor called *LAF2AWAX*. The parameter set can be consulted in Appendix 4.

- ***Total leakage reactance estimation results:***

Section 6.2.2 introduced the estimation of the total leakage inductance and its expression given by *(6.2.4)*. Since in the simulation program all quantities are in p.u., being referred to base values, the total leakage inductance would be a very small quantity when expressed in p.u., in the range of *0.0003-0.0004*. Therefore it was seen more appropriate to estimate the total leakage reactance.

Results for total leakage reactance estimation are presented in *Table 6.3.1*, in which: V is the d.c. link voltage, $h=0.4\,\mu s$ is the integration step, $Tc=200\ \mu s$ is the sampling step, *ualfa* is the voltage step applied to the motor and u_{dc} is the d.c. voltage in p.u. (i.e. the ratio of the d.c. link voltage to the base voltage). To facilitate the estimation procedures

two files written in Pascal language are used: main file *LSIGMA.pas* implements the estimation procedure and *ASMOTOR2* unit contains data for the tested motor.

Sim. 788 - 28/10/1997

input data	***files***	***results***
V=280 V *h=0.4 μs* *Tc= 200 μs* *0- 200 steps* *ualfa=(2/3)· u_{dc}* *u_{dc} =280/(165)=1.6922* *xsig (expected)=0.2015*	*LSIGMA.pas* *unit ASMOTOR2*	*t2 =0.0104* *ialf2 =1.3370* *t3 =0.0190* *ialf3 =0.0772* *t4 =0.0194* *ialf4 =-1.2640* *xsig_est=0.2114* *error=4.91%*

Table 6.3.1- Simulation results for x_σ estimation

The simulation results described and presented in *Table 6.3.1* are plotted in *Figure 6.3.1.*

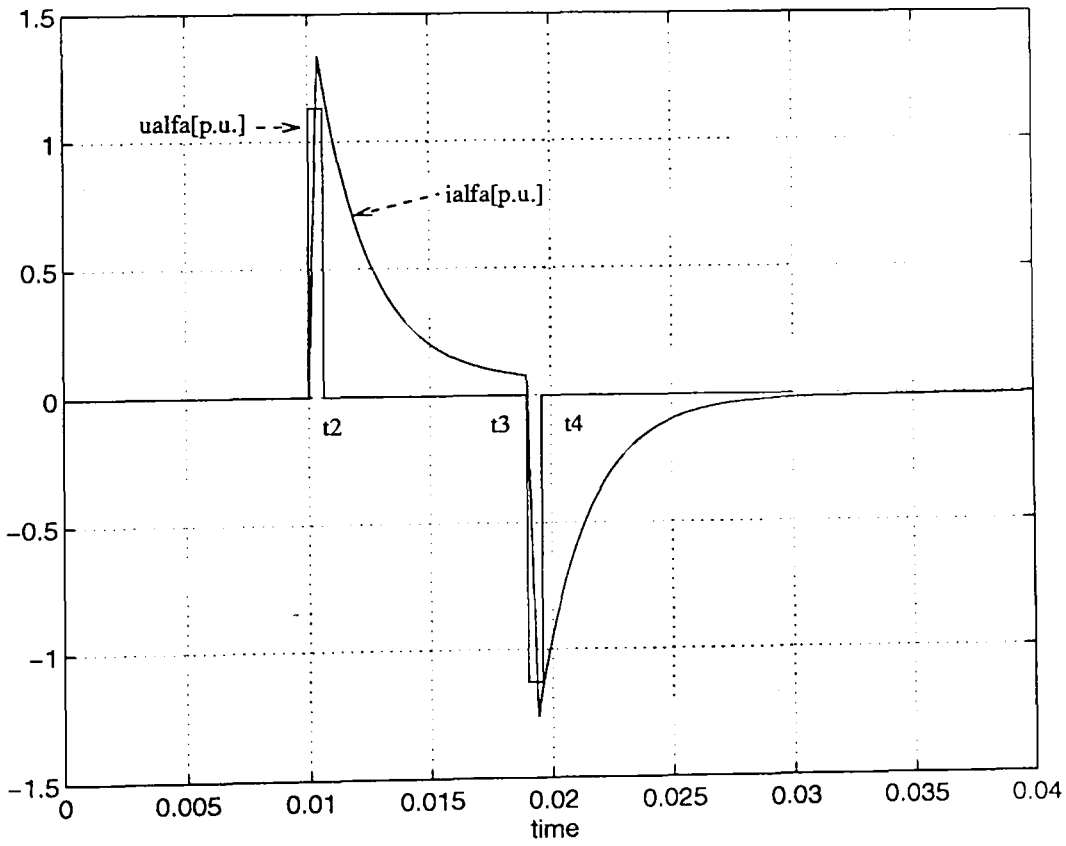

Figure 6.3.1 - The voltage impulses applied to the motor and current response (Sim 788)

At time *t2* the motor is short-circuited for the first time. As seen in *Figure 6.3.1*, the first voltage impulse is applied at *0.01 seconds*. The corresponding current at the moment *t2* is *ialf2* . At time *t3* the negative impulse is applied and *ialf3* is the corresponding current. The motor is then short-circuited for a second time at *t4*, and remains in that condition. The current at *t4* is *ialf4*. In the column *'results'*, all currents are in p.u. and all time quantities are in seconds.

The estimated total leakage reactance, named *xsig_est* , is calculated using:

$$xsig\_est = \frac{2}{3} u_{dc} \frac{t4 - t3}{ialf3 - ialf4} 2\pi F_{no} \qquad (6.3.1)$$

where *Fno* is the nominal frequency of the motor.

Table 6.3.2 and *Table 6.3.3* present further simulation results. In *Table 6.3.2*, by using a larger sampling step, a slight increase of the estimation error from *4.9%* to *5.4%* is registered.

Sim. 790 - 28/10/1997

input data	***files***	***results***
V=280 V *h=0.4 μs* *Tc= 426.66 μs* *0- 100 steps* *ualfa=2/3 u_{dc}* *xsig (expected)=0.2015*	*LSIGMA.pas* *unit ASMOTOR2*	*t2 =0.0107* *ialf2 =1.4159* *t3 =0.0192* *ialf3 =0.0830* *t4 =0.0196* *ialf4 =-1.3377* *xsig_est=0.2125*

Table 6.3.2- Simulation results for x_σ estimation

For *Table 6.3.3* simulation, *ualfa* is changed to one third of the *d.c.* voltage (in p.u.) and *Figure 6.3.2* shows voltage and current graphs for *Table 6.3.3* simulation.

Sim. 795 - 28/10/1997

input data	***files***	***results***
V=280 V *h=0.4 μs* *Tc= 426.66 μs* *0- 100 steps* *ualfa=1/3 u_{dc}* *xsig (expected)=0.2015*	*LSIGMA.pas* *unit ASMOTOR2*	*t2 =0.0111* *ialf2 = 1.2906* *t3 =0.0192* *ialf3 =0.0859* *t4 =0.0200* *ialf4 =-1.2144* *xsig_est=0.2322*

Table 6.3.3- Simulation results for x_σ estimation

Another simulation when *ualfa* is one third of the *d.c.* voltage and sampling step is *200 μs* is presented in *Table 6.3.4*. It is seen that estimation errors are between *9.3%* and *15.2%* for *ualfa=1/3 u_{dc}* . When this value of *ualfa* is applied to the motor terminals, the current takes a longer time to arrive at a near peak value. This interval may be longer then

the time rotor constant, therefore the assumption that lead to *(6.2.3)* equation is no longer valid. Thus the occurrence of increasing errors.

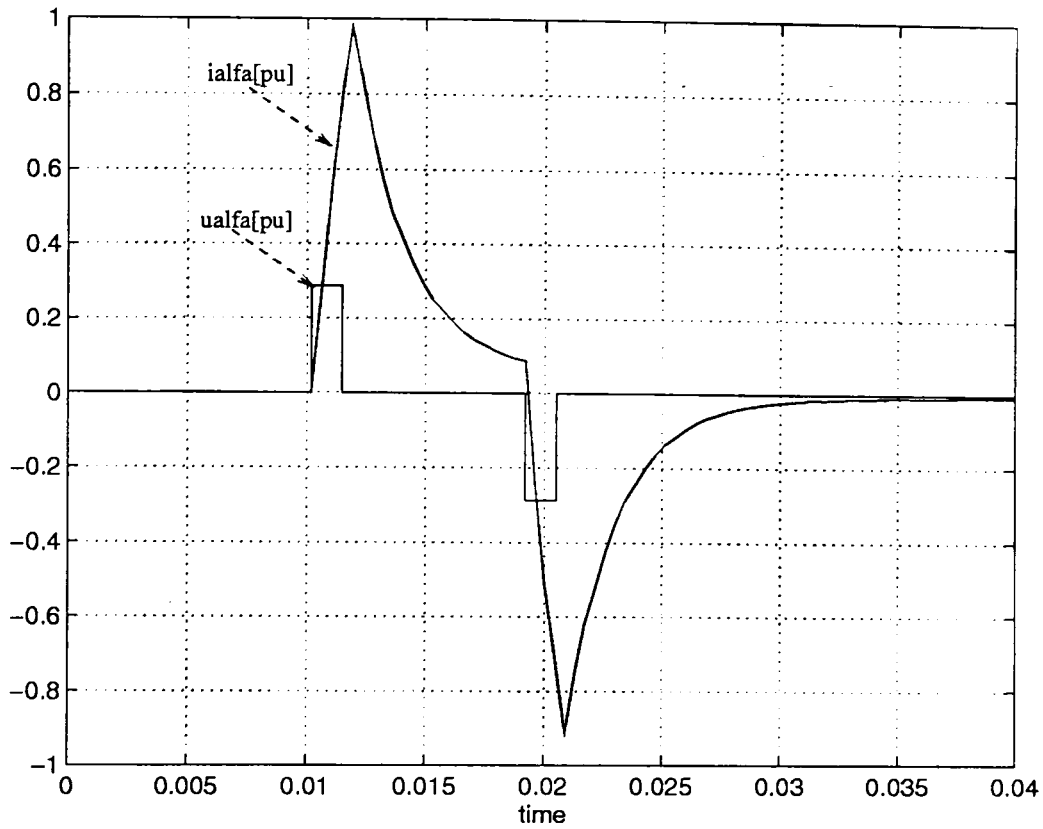

Figure 6.3.2 - The voltage impulses applied to the motor and current response

input data	*files*	*results*
V=280 V	*LSIGMA.pas*	*t2 =0.0106*
h=0.4 μs	*unit ASMOTOR2*	*ialf2 =0.9609*
Tc= 200 μs		*t3 =0.0190*
0- 200 steps		*ialf3 =0.0588*
*ualfa=1/3*u_{dc}*		*t4 =0.0196*
		ialf4 =-0.9068
xsig (expected)=0.2015		*xsig_est=0.2202*
		error=9.3%

Table 6.3.4- Simulation results for x_σ estimation

- **Stator resistance estimation results :**

Estimation of the stator resistance is carried out using the formula in *(6.2.6)*. A first set of results is presented in *Table 6.3.5*, where: *i_imp*= 0.25 is the current applied in the first two intervals: *0.25 p.u.* and respectively *-0.25 p.u.* and *iimp_c*= 0.8 is the *(c)* interval current .

In the results column: *usalm_a* is the average voltage of the *(a)* interval, *u_cf* is the voltage at the end of *(c)* interval, *isalm_a* is the average current over the *(a)* interval, *isalm_c* is the average current over the *(c)* interval and *rs_stim* is the estimated stator resistance.

Sim. 799 - 28/10/1997

input data	***files***	***results***
V=280 V	*ST_RES.pas*	*usalm_a =0.0187*
h=0.4 µs	*unit ASMOTOR2*	*u_cf =0.0598*
Tc= 426.66 µs		*isalm_c =0.7983*
0- 16 000 steps		*isalm_a =0.2494*
i_imp= 0.25 p.u.		
iimp_c= 0.8 p.u.		
r_s (expected)=0.0745		*rs_stim=0.0749*

Table 6.3.5- Simulation results for r_s estimation

The results presented in *Table 6.3.5* are plotted in the *Figure 6.3.3*.

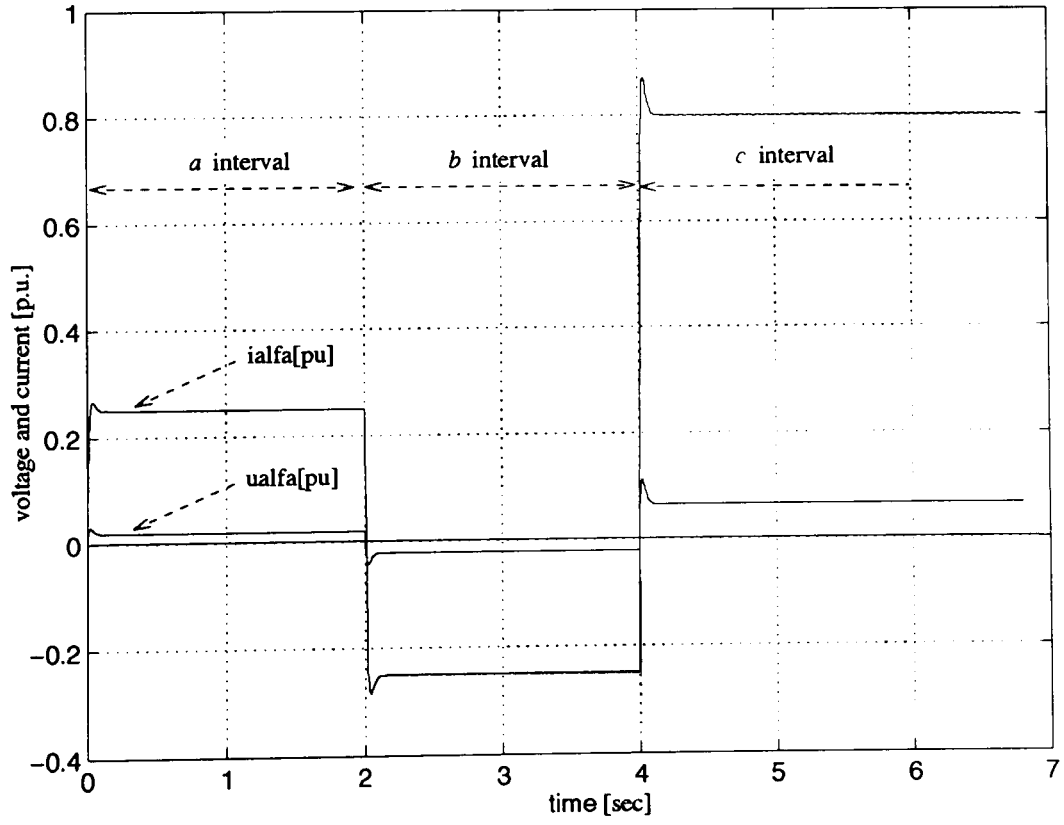

Figure 6.3.3 - The voltage impulses applied to the motor and current response

Another set of results, for a sampling step of *200 µs*, can be seen in *Table 6.3.6*. Good results exhibiting errors of only *0.5%* to *0.9%* are obtained in estimating the stator resistance for any of the two sampling steps used.

Sim. 800 - 28/10/1997

input data	*files*	*results*
V=280 V *h=0.4 μs* *Tc= 200 μs* *0- 30 000 steps* *i_imp= 0.25* *iimp_c= 0.8* *r_s (expected)= 0.0745*	*ST_RES.pas* *unit ASMOTOR2*	*usalm_a =0.0193* *u_cf =0.0598* *isalm_c =0.7973* *isalm_a =0.2494* *rs_stim=0.0738*

Table 6.3.6- Simulation results for r_s estimation

- **<u>Rotor resistance estimation results :</u>**

Rotor resistance is estimated using the previously calculated stator resistance and the formula of *(6.2.8)*. Thus *rs= 0.0749* is part of the input data. *MaxUsaRef* is the maximum voltage at the beginning of the *(c)* interval and the corresponding current is *i_maxUsaRef*. The average current over the *(b)* interval is denoted by *im_b*. With these notations the formula *(6.2.8)* becomes :

$$rr\_stim = \frac{MaxUsaRef - rs \cdot i\_maxUsaRef}{i\_maxUsaRef - im\_b} \qquad (6.3.2)$$

Simulation results are presented numerically in *Table 6.3.7* and graphically in *Figure 6.3.4*.

Sim. 802 - 28/10/1997

input data	*files*	*results*
V=280 V *h=0.4 μs* *Tc= 426.66 μs* *9 300- 10 000 steps* *i_imp= 0.25 pu iimp_c= 0.8 pu* *rs= 0.0749* *r_r (expected)=0.0781*	*ROT_RS2.pas* *unit ASMOTOR2*	*MaxUsaRef = 0.1014* *i_maxUsaRef= 0.7723* *im_b = -0.2488* *rr_stim= 0.0754* *error=3.46%*

Table 6.3.7- Simulation results for r_R estimation

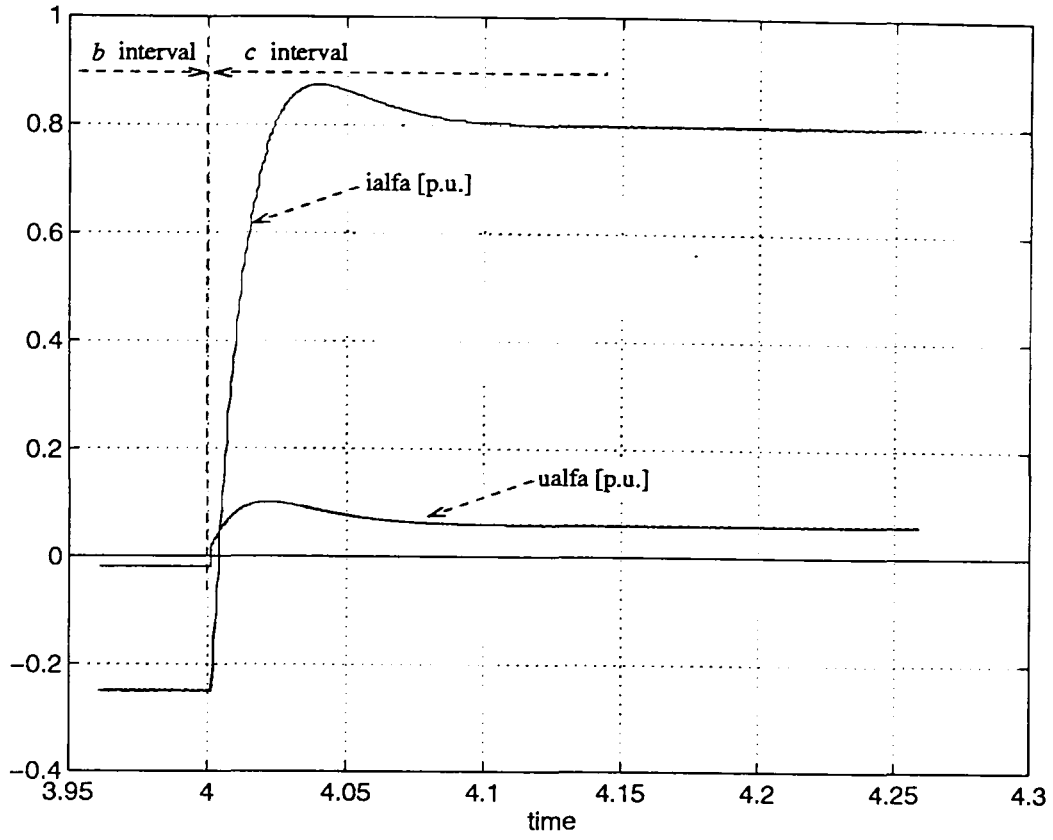

Figure 6.3.4 - The voltage and current curves at the beginning of the c interval

A further set of estimated results for the rotor resistance, using a $200\,\mu s$ sampling step, is shown in *Table 6.3.8.* Good results are obtained in estimating the rotor resistance with errors of around *3.3%*.

Sim. 803 - 28/10/1997

input data	***files***	***results***
V=280 V *h=0.4 µs* *Tc= 200 µs* *19 000- 22 000 steps* *i_imp= 0.25* *iimp_c= 0.8* *rs= 0.0738* *r_r (expected)=0.0781*	*ROT_RS2.pas* *unit ASMOTOR2*	*MaxUsaRef = 0.1006* *i_maxUsaRef= 0.7620* *im_b = -0.2487* *rr_stim= 0.0756*

Table 6.3.8- Simulation results for r_R estimation

- **Rotor time constant estimation results:**

The steps forming the basis of the algorithm used for estimating the rotor time constant has been reported in *Section 6.2.3*, where *Figure 6.2.8* graphically explains the appropriate variables. The estimated rotor time constant, *Tr_stim* is obtained from:

$$Tr\_stim = timp\_UsaRef67 - timp\_minUsaRef \qquad (6.3.3)$$

Tables *6.3.9* and *6.3.10* present the test parameters and the values of rotor time constant calculated for sampling periods of *426* μs and *200* μs.

Sim. 804 - 28/10/1997

input data	***files***	***results***
V=280 V *h=0.4* μs *Tc= 426* μs *4 500- 5 500 steps* *i_imp= 0.25* *iimp_c= 0.8* *T_r (expected)=0.0234*	*ROT_CT2.pas* *unit ASMOTOR2*	*MaxUsaRef = -0.0185* *MinUsaRef = -0.0386* *UsaRef67 = -0.0259* *timp_minUsaRef= 2.0218* *timp_UsaRef67= 2.0529* *Tr_stim=0.0311*

Table 6.3.9- Simulation results for T_R *estimation*

Sim. 8xx - 28/10/1997

input data	***files***	***results***
V=280 V *h=0.4* μs *Tc= 200* μs *9 700- 11 000 steps* *i_imp= 0.25 iimp_c= 0.8* *T_r (expected)=0.0234*	*ROT_CT2.pas* *unit ASMOTOR2*	*MaxUsaRef = -0.0192* *MinUsaRef = -0.0380* *UsaRef67 = -0.0261* *timp_minUsaRef= 2.0216* *timp_UsaRef67= 2.0528* *Tr_stim=0.0312*

Table 6.3.10- Simulation results for T_R *estimation*

The corresponding graphical output is shown in *Figure 6.3.5* and *Figure 6.3.6* . In both cases, for two different values of the sampling step, the estimate of the rotor time constant is found to be in error by about *30%*. This is thought to be due to error in considering the exponential curve of the terminal voltage.

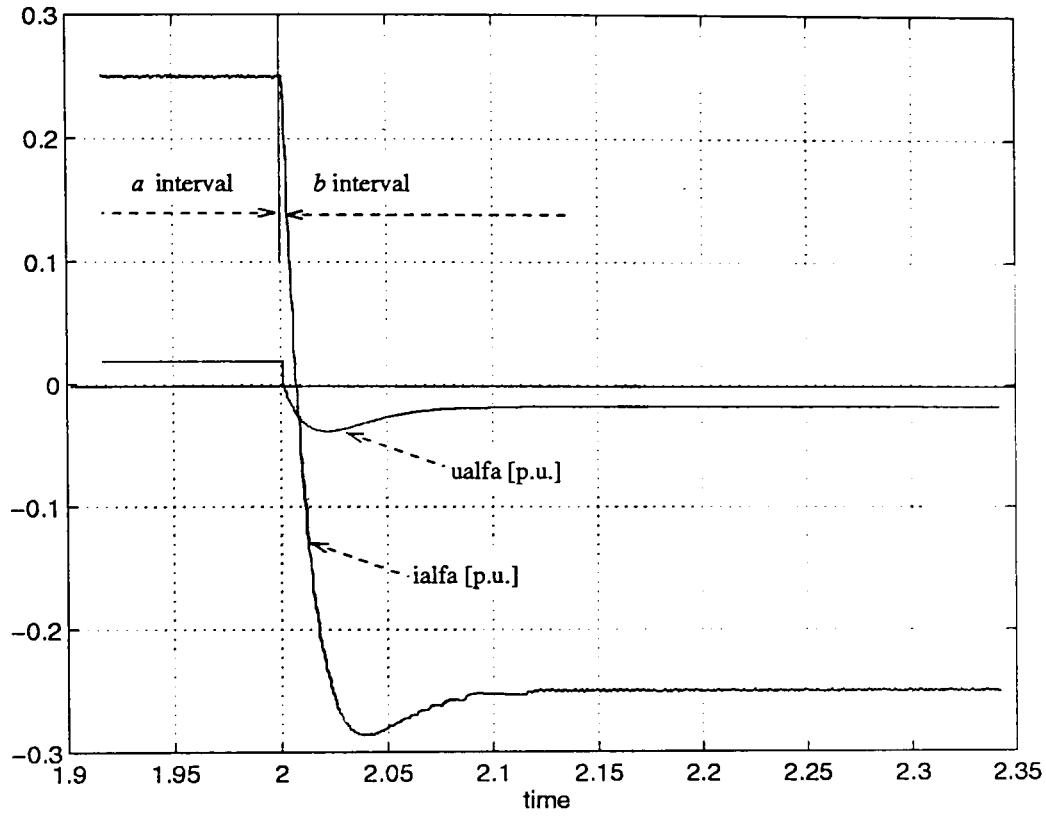

Figure 6.3.5 - The voltage and current curves at the beginning of the b interval

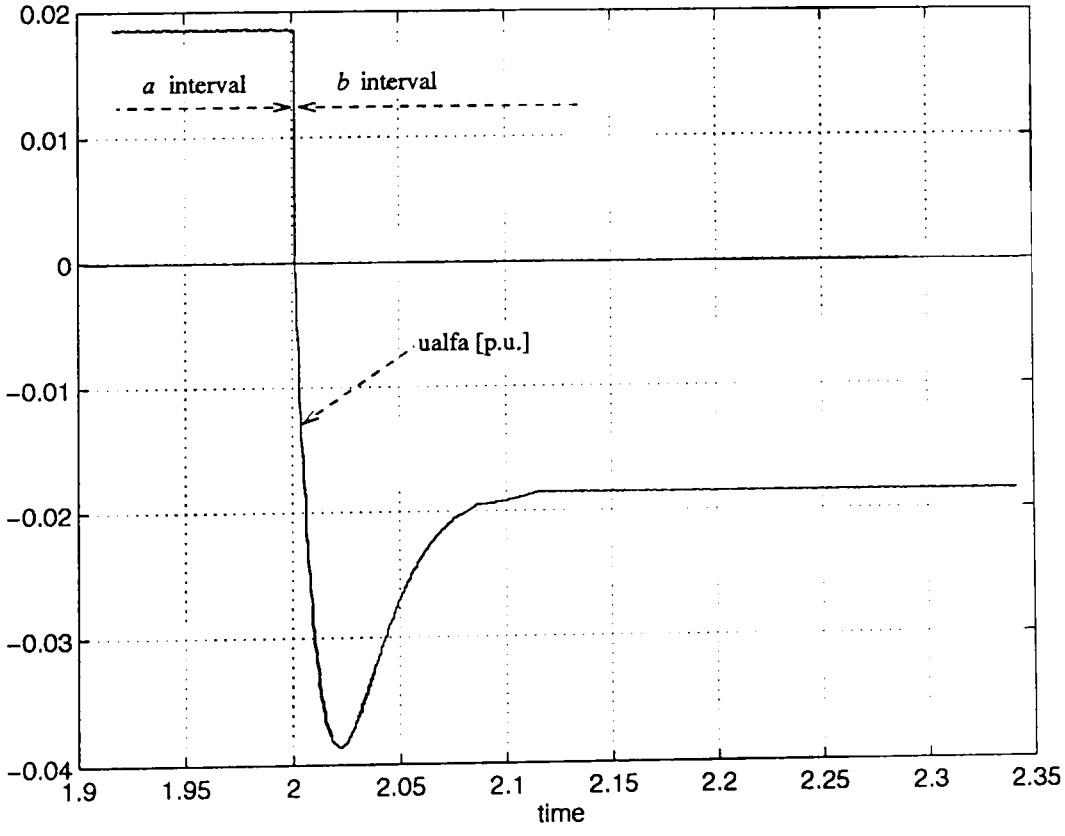

Figure 6.3.6 - The voltage curve at the beginning of the b interval

In *Section 6.2* it has been shown that the rotor flux in *(b)* interval changes in an exponential manner and that the time constant of this curve is actually the rotor time

constant. However some ideal conditions have been assumed by neglecting magnetic saturation, eddy current effects and current displacement. Additionally, when digitally implementing the controller, the output voltage is not smooth and therefore a fitted curve may be necessary.

Corresponding to the results of *Table 6.3.9* , the voltage again is plotted in *Figure 6.3.7*. On the same graph two curves of different time constant, $\tau_1 = 0.02$ and $\tau_2 = 0.025$ are plotted.

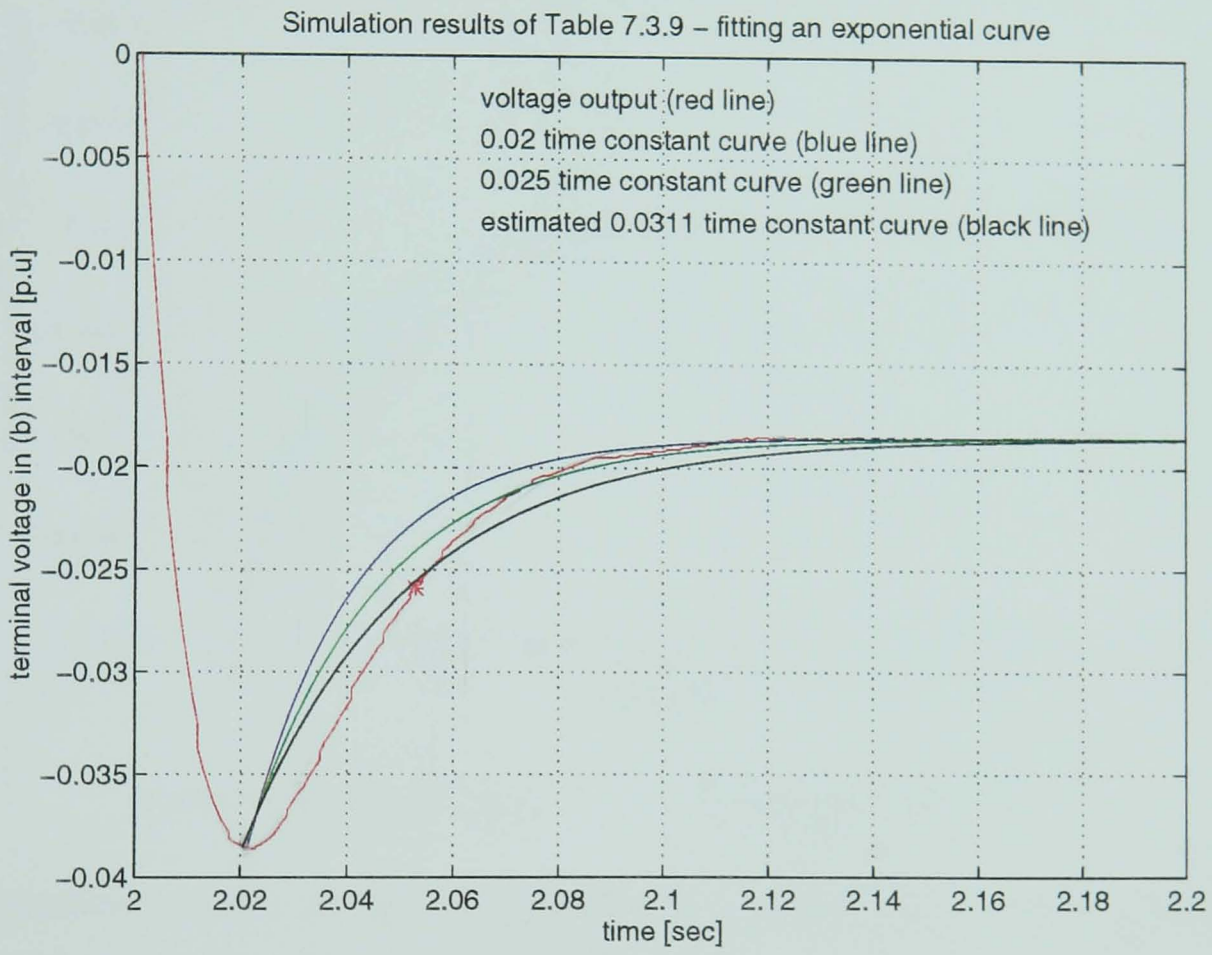

Figure 6.3.7 - Exponential curves to fit the terminal voltage curve

It is also shown the curve defined by the estimated rotor time constant, *Tr_stim=0.0311*, taken from results in *Table 6.3.9*. However it does not fit well on the voltage curve. It can be considered that the curve defined by $\tau_1 = 0.02$ gives a good approximation of the voltage curve. A new value of *timp_UsaRef67* can be graphically extracted from the plots and that is: *timp_UsaRef67=2.0413 s*. According to *(6.3.3)* and knowing that *timp_minUsaRef= 2.0218* then a new *Tr_stim= 0.0195* is obtained. Consequently the estimation error has decreased from *32.9%* to *16.6%*. It is appreciated that by finding a better suited curve, the error may be further diminished.

The same procedure is applied to results of *Table 6.3.10.* Plots are shown in *Figure 6.3.8.* A curve defined by $\tau = 0.022$ is chosen and new value of *timp_UsaRef67* is *2.0439.* Thus *Tr_stim=0.0223* and the estimation error equals *4.7%.*

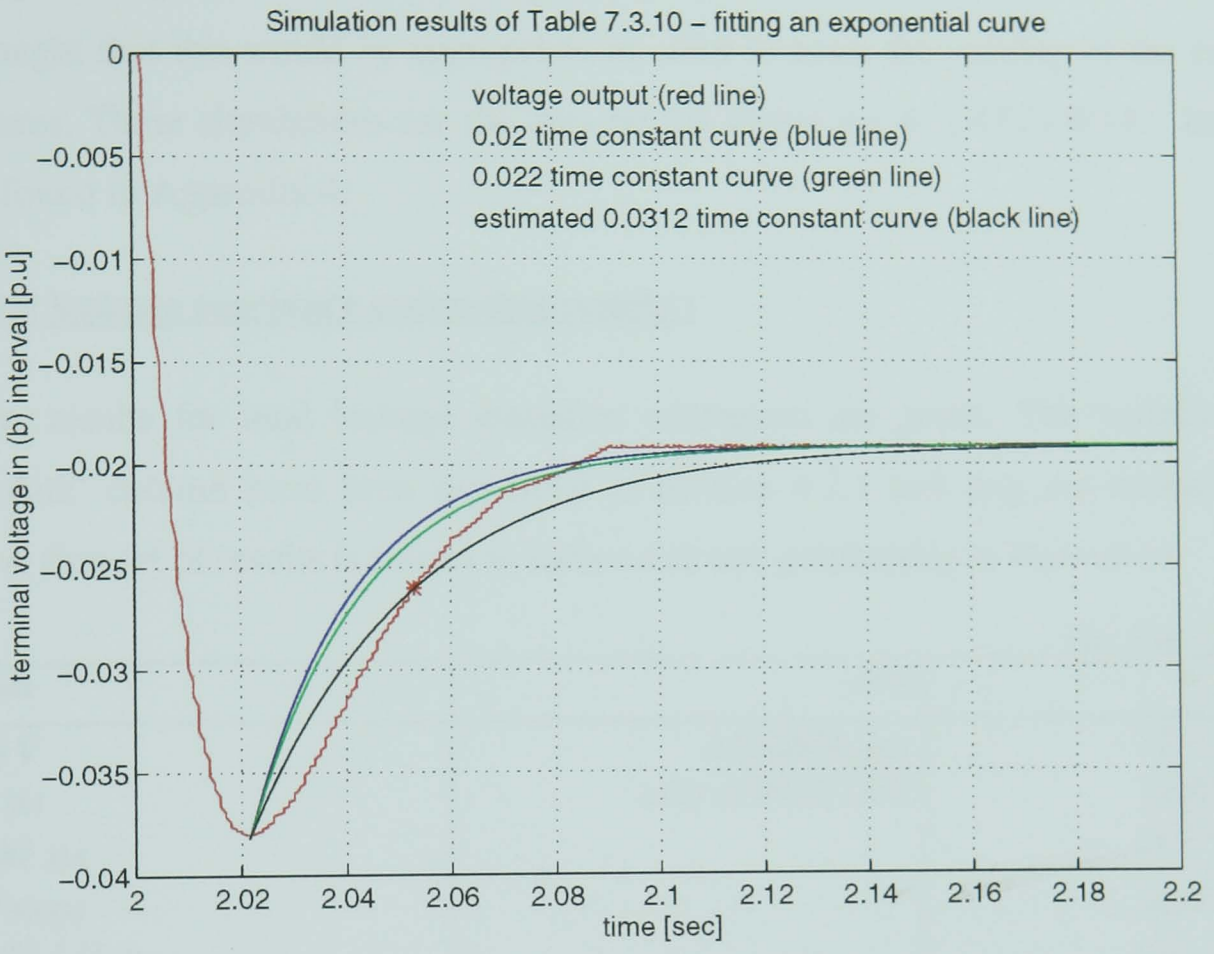

Figure 6.3.8 - Exponential curves to fit the terminal voltage curve

Next subsection presents similar simulations when applied to a different motor. The parameter set used is of motor no. 4. [Appendix 4]

6.3.2 *Simulation Results - part II -*

Similarly to *Section 6.3.1*, this section presents simulation results for a different motor. It was thought that this would be appropriate in order to asses the validity of the estimation procedures. These simulations use the data set for motor no. 4, *LAF1AWAX* . Its data set can be found in Appendix 4.

- **Total leakage reactance estimation results :**

Initially, results for total leakage reactance estimation are given. The variables in the `Input data` column have been explained in *Section 6.3.1* and *xsig_est* estimation uses *(6.3.1)*. A first set of results is shown in *Table 6.3.11* and graphically in *Figure 6.3.9*.

Sim. 748 - 28/10/1997

input data	***files***	***results***
V=280 V *h=0.4 μs* *Tc= 200 μs* *0- 200 steps* *ualfa=(2/3)·u_{dc}* *xsig (expected)=0.1717*	*LSIGMA.pas* *unit ASMOTOR1*	*t2 =0.0106* *ialf2 =0.8622* *t3 =0.0190* *ialf3 =0.0547* *t4 =0.0196* *ialf4 =-0.8121* *xsig_est=0.1872* *error= 9.03%*

Table 6.3.11- Simulation results for x_σ estimation

The following tables give results for different sampling periods or for different value of the voltage pulse applied to the motor.

Sim. 749 - 28/10/1997

input data	***files***	***results***
V=280 V *h=0.4 μs* *Tc= 426.66 μs* *0- 100 steps* *ualfa=(2/3)·u_{dc}* *xsig (expected)=0.1717*	*LSIGMA.pas* *unit ASMOTOR1*	*t2 =0.0111* *ialf2 =1.1621* *t3 =0.0192* *ialf3 =0.0801* *t4 =0.0200* *ialf4 =-1.0917* *xsig_est=0.1966* *error= 14.5%*

Table 6.3.12- Simulation results for x_σ estimation

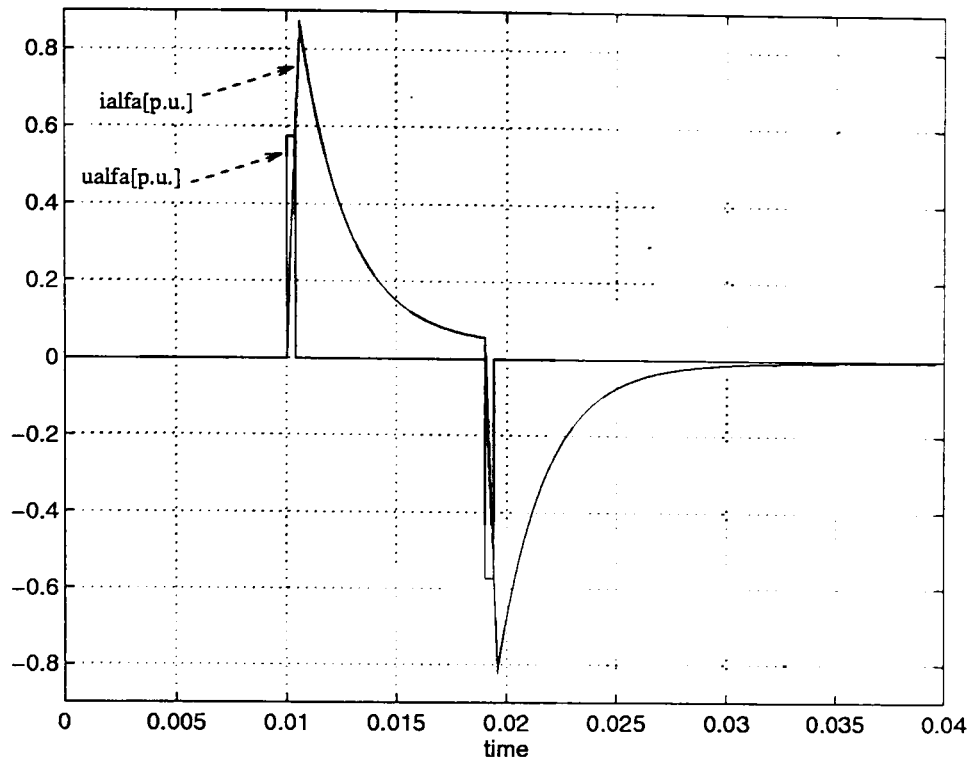

Figure 6.3.9 - The voltage pulses and the current response(Table 6.3.11)

Sim. 750 - 28/10/1997

input data	***files***	***results***
V=280 V *h=0.4 μs* *Tc= 426.66 μs* *0- 100 steps* *ualfa=(1/3)·u_{dc}* *xsig (expected)=0.1717*	*LSIGMA.pas* *unit ASMOTOR1*	*t2 =0.0119* *ialf2 =0.9627* *t3 =0.0192* *ialf3 =0.0869* *t4 =0.0209* *ialf4 =-0.9156* *xsig_est=0.2298* *error= 33.8%*

Table 6.3.13- Simulation results for x_σ estimation

Sim. 751 - 28/10/1997

input data	***files***	***results***
V=280 V *h=0.4 μs* *Tc= 200 μs* *0 - 200 steps* *ualfa=(1/3)·u_{dc}* *xsig (expected)=0.1717*	*LSIGMA.pas* *unit ASMOTOR1*	*t2 =0.0114* *ialf2 =0.8572* *t3 =0.0190* *ialf3 =0.0686* *t4 =0.0204* *ialf4 =-0.9015* *xsig_est=0.2176* *error= 26.7%*

Table 6.3.14- Simulation results for x_σ estimation

The best estimation by an error *9%* is obtained using a sampling time of *200* μs and a voltage impulse $ualfa=(2/3)\cdot u_{dc}$. This is similar to the results of *Section 6.3.1.* where lowest error of *5.46%* is obtained for the same set of input data.

- **Stator resistance estimation results :**

The stator resistance uses *(6.2.6)* and if this expression is rewritten with notation from the `*results*` column it becomes:

$$rs\_stim = \frac{u\_cf - usalm\_a}{isalm\_c - isalm\_a} \qquad (6.3.4)$$

For the estimate of stator resistance only two simulations are presented and results are given in *Tables 6.3.15* and *6.3.16.* Corresponding to *Table 6.3.15* results, a plot is given in *Figure 6.3.10.*

Sim. 757 - 28/10/1997

input data	***files***	***results***
V=280 V *h=0.4 μs* *Tc= 426.66 μs* *0- 16 000 steps* *i_imp= 0.25* *iimp_c= 0.8* *r_s (expected)=0.0813*	*ST_RES.pas* *unit ASMOTOR1*	*usalm_a =0.0204* *u_cf =0.0650* *isalm_c =0.7982* *isalm_a =0.2494* *rs_stim= 0.0811* *error= 0.25%*

Table 6.3.15- Simulation results for r_S estimation

Sim. 760 - 28/10/1997

input data	***files***	***results***
V=280 V *h=0.4 μs* *Tc= 200 μs* *i_imp= 0.25* *iimp_c= 0.8* *r_s (expected)=0.0813*	*ST_RES.pas* *unit ASMOTOR1*	*usalm_a =0.0203* *u_cf =0.0650* *isalm_c =0.7973* *isalm_a =0.2494* *rs_stim= 0.0813*

Table 6.3.16- Simulation results for r_S estimation

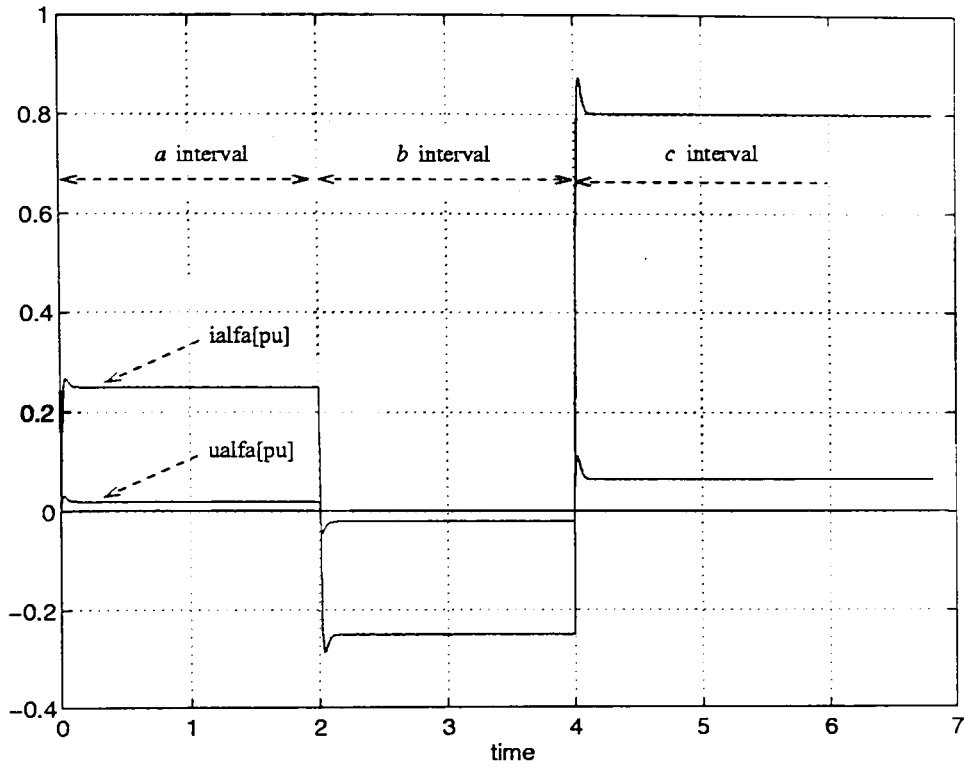

Figure 6.3.10 - The voltage impulses applied to the motor and current response (Table 6.3.15)

Both cases presented show very good results with an error of only *0.2%* in the case of *426.66 μs* sampling time. This shows once more the accordance with the results for stator resistance given in *Section 6.3.1.*

- **Rotor resistance estimation results :**

The rotor resistance is estimated with the formula derived from *(6.2.8)* :

$$rr\_stim = \frac{MaxUsaRef - rs \cdot i\_maxUsaRef}{i\_maxUsaRef - im\_b} \qquad (6.3.5)$$

As the expression contains *rs* , this estimation is done after the *rs* evaluation and then *rs= 0.0811* is part of the input data. Two sets of results are given in the following *Tables 6.3.17* and *6.3.18.*

Sim. 762 - 28/10/1997

input data	*files*	*results*
V=280 V *h=0.4 μs* *Tc= 426.66 μs* *i_imp= 0.25* *iimp_c= 0.8* *rs= 0.0811* *r_r (expected)=0.0857*	*ROT_RS2.pas* *unit ASMOTOR1*	*MaxUsaRef = 0.1122* *i_maxUsaRef= 0.7740* *im_b = -0.2488* *rr_stim= 0.0860* *error= 0.35%*

Table 6.3.17- Simulation results for r_R estimation

Sim. 770 - 28/10/1997

input data	*files*	*results*
V=280 V *h=0.4 μs* *Tc= 200 μs* *19 000- 22 000 steps* *i_imp= 0.25* *iimp_c= 0.8* *rs= 0.0813* *r_r (expected)=0.0857*	*ROT_RS2.pas* *unit ASMOTOR1*	*MaxUsaRef = 0.1123* *i_maxUsaRef= 0.7676* *im_b = -0.2487* *rr_stim= 0.0859*

Table 6.3.18- Simulation results for r_R estimation

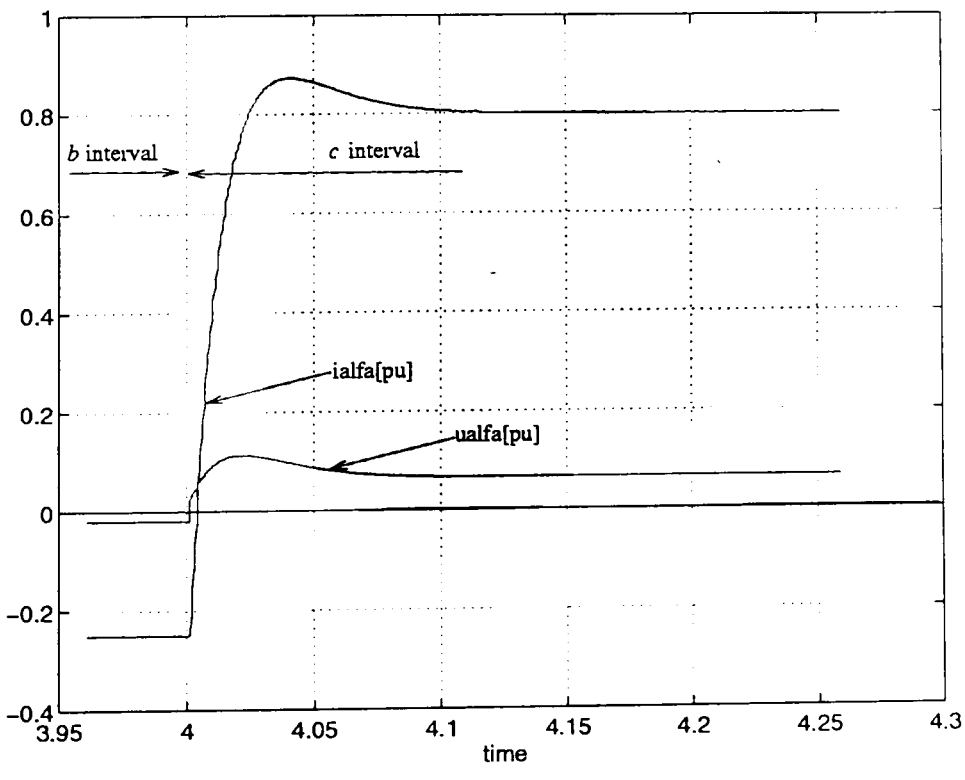

Figure 6.3.11 - The voltage and current curves at the beginning of the c interval (Table 6.3.17)

Results obtained for rotor resistance are satisfactory. Different sampling time and different levels of current are used, and errors encountered are around *0.35%*. This error has the same range as the error for the motor no. 3 results [Section 6.3.1].

- **Rotor time constant estimation results :**

With data from '*results*' column, the rotor time constant is estimated using:

$$Tr\_stim = timp\_UsaRef67 - timp\_minUsaRef \qquad (6.3.6)$$

Results are given in *Tables 6.3.19* and *6.3.20* for two sampling periods and voltage and current curves are plotted in *Figure 6.3.12.*

Sim. 780 - 28/10/1997

input data	***files***	***results***
V=280 V *h=0.4 µs* *Tc= 426 µs* *4 500- 5 500 steps* *i_imp= 0.25* *iimp_c= 0.8* *T_r (expected)=0.0249*	*ROT_CT2.pas* *unit ASMOTOR1*	*MaxUsaRef = -0.0202* *MinUsaRef = -0.0428* *UsaRef67 = -0.0285* *timp_minUsaRef= 2.0218* *timp_UsaRef67= 2.0546* *Tr_stim= 0.0328*

Table 6.3.19- Simulation results for T_R estimation

Sim. 783 - 28/10/1997

input data	***files***	***results***
V=280 V *h=0.4 µs* *Tc= 200 µs* *9 700- 11 000 steps* *i_imp= 0.25* *iimp_c= 0.8* *T_r (expected)=0.0249*	*ROT_CT2.pas* *unit ASMOTOR1*	*MaxUsaRef = -0.0201* *MinUsaRef = -0.0424* *UsaRef67 = -0.0283* *timp_minUsaRef= 2.0218* *timp_UsaRef67= 2.0554* *Tr_stim= 0.0336*

Table 6.3.20 - Simulation results for T_R estimation

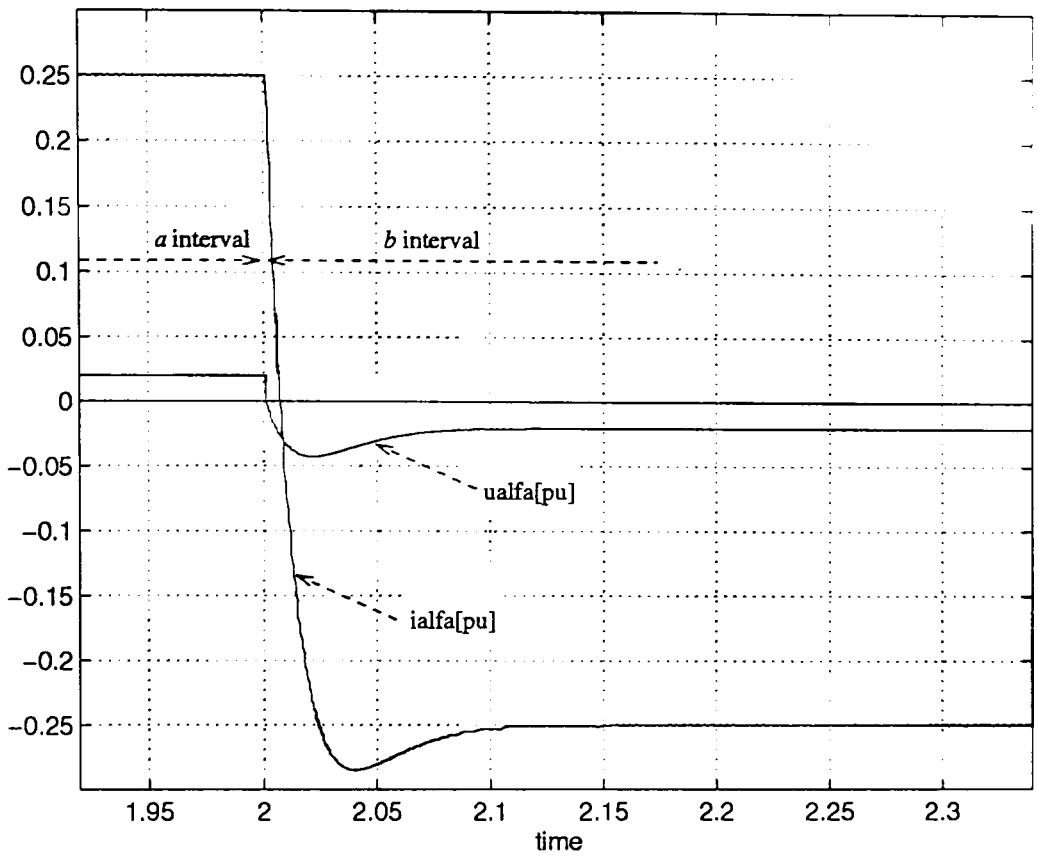

Figure 6.3.12 - The voltage and current curves at the beginning of the b interval (Table 6.3.19)

Rotor time constant is estimated with an error between *31* and *34%*. It leads to the conclusion that the procedure for rotor resistance estimation is faulty or the extraction of needed data from the voltage curves is done erroneous.

It has been reasoned in *Section 6.3.1* that it may be appropriate to fit a curve on the voltage output and then to estimate the rotor time constant. This approach is also followed in this case. Replotting voltage results of *Table 6.3.19* and finding two curves that may be considered suited is done in *Figure 6.3.13*.

From the curve defined by $\tau = 0.025$, a new value of *timp_UsaRef67= 2.04625* can be deduced and *Tr_stim = 0.02445* is re-evaluated. Hence the error lowers to *1.8%*.

Similarly is done with results of the *200* μs sampling period. Curves defined by *0.02* and *0.025* time constants are used. In this case, from the curve defined by $\tau = 0.025$, a new value of *timp_UsaRef67= 2.0459* can be deduced and *Tr_stim = 0.0241* is re-evaluated. Error reduces to *3.2%*. The corresponding figure is *Figure 6.3.14*.

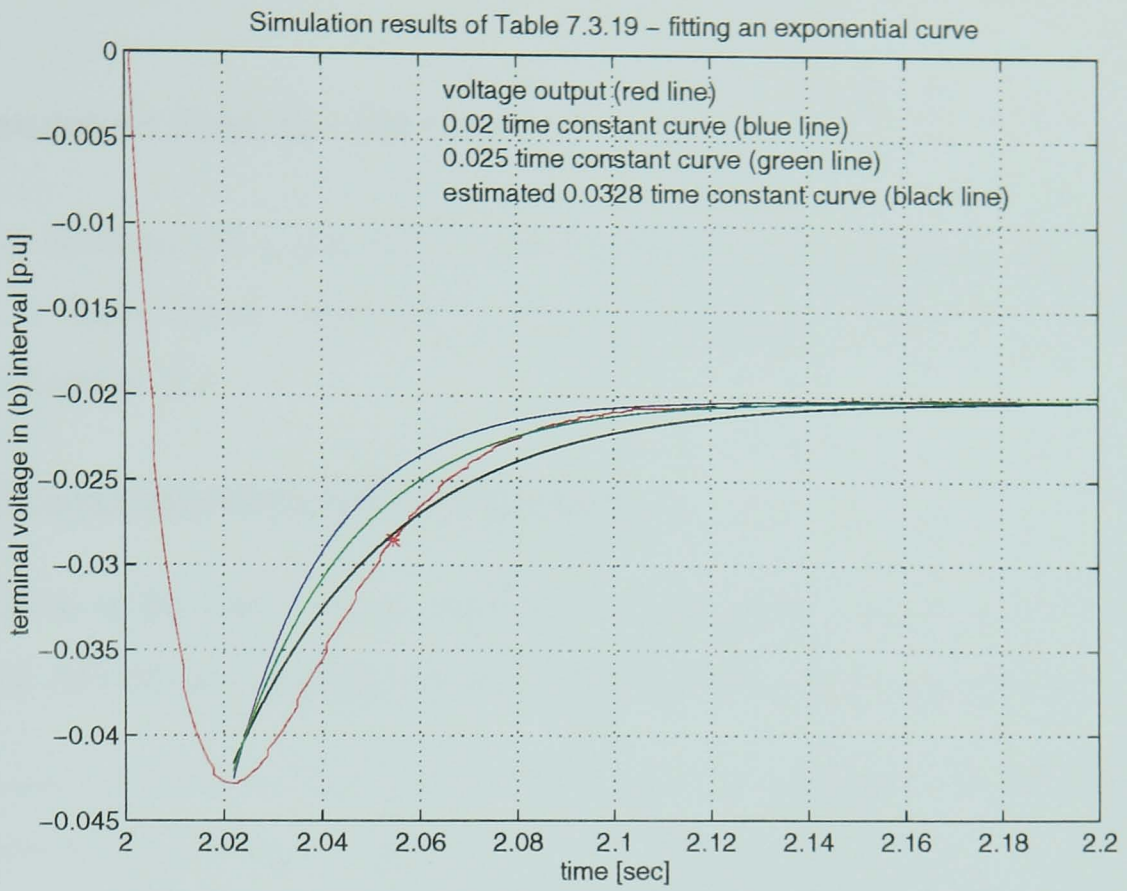

Figure 6.3.13 - Fitting exponential curves on the voltage curve

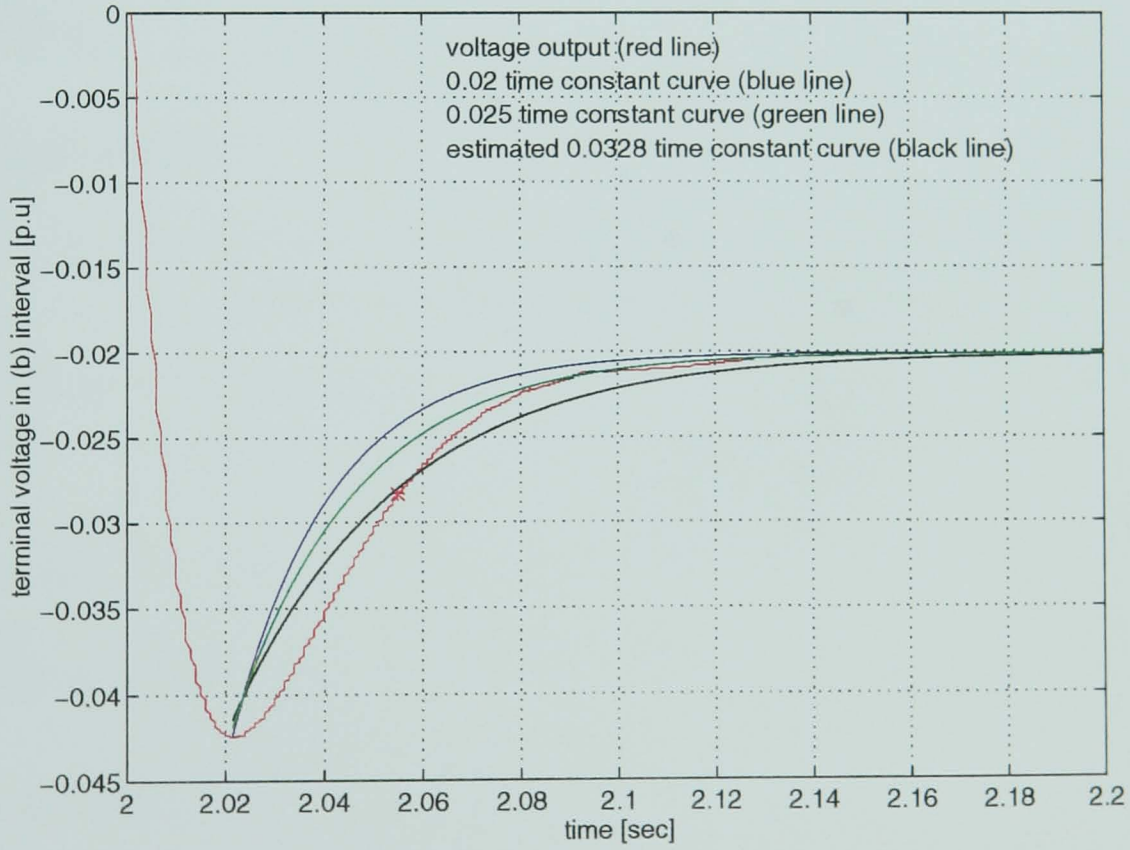

Figure 6.3.14 - Fitting exponential curves on the voltage curve

A third part of simulation results follows. The data set of motor no. 1 , *Lafert* is used. [Appendix 4].

6.3.2 *Simulation Results - part III -*

Similarly to *Sections 6.3.1* and *6.3.2*, this section presents simulation results for another motor, no. 1 *Lafert* from Appendix 4. This is the motor utilised for the practical tests reported in *Section 6.4.*

- **Total leakage reactance estimation results :**

The estimation of the total leakage reactance, *xsig_est* uses *(6.3.1)*. Two sets of results for *200* μs and *426.66* μs sampling periods are presented in *Table 6.3.21* and *6.3.22.*

Sim. 795 - 30/10/1997

input data	*files*	*results*
V=280 V *h=0.4* μs *Tc= 200* μs *0- 200 steps* *ualfa=(2/3)·u_{dc}* *xsig (expected)=0.2132*	*LSIGMA.pas* *unit ASLAFERT*	*t2 =0.0112* *ialf2 =0.9175* *t3 =0.0190* *ialf3 =0.0885* *t4 =0.0202* *ialf4 =-0.8507* *xsig_est=0.2408* *error= 12.94%*

Table 6.3.21 - Simulation results for x_σ estimation

Sim. 797 - 30/10/1997

input data	*files*	*results*
V=280 V *h=0.4* μs *Tc= 426.66* μs *0- 100 steps* *ualfa=(2/3)·u_{dc}* *xsig (expected)=02132.*	*LSIGMA.pas* *unit ASLAFERT*	*t2 =0.0115* *ialf2 =0.9650* *t3 =0.0192* *ialf3 =0.0964* *t4 =0.0204* *ialf4 =-0.8936* *xsig_est=0.2433* *error= 14.10%*

Table 6.3.22 - Simulation results for x_σ estimation

The best estimation by an error *12.93%* is obtained using a sampling time of *200* μs and a voltage impulse *ualfa=(2/3)·u_{dc}*. This is similar to the results of *Section 6.3.1.* where the lowest error is obtained for the same set of input data.

- **Stator resistance estimation results :**

For the estimate of stator resistance two simulations are presented and results are given in *Tables 6.3.23* and *6.3.24.* Both cases presented show very good results with an error of *1.07%.*

Sim. 810- 30/10/1997

input data	*files*	*results*
V=280 V *h=0.4 µs* *Tc= 426.66 µs* *0- 16 000 steps* *i_imp= 0.25 iimp_c= 0.8* *r_s (expected)=0.1224*	*ST_RES.pas* *unit ASLAFERT*	*usalm_a =0.0313* *u_cf =0.0978* *isalm_c =0.7980* *isalm_a =0.2493* *rs_stim= 0.1211* *error= 1.07%*

Table 6.3.23 - Simulation results for r_S estimation

Sim. 811 - 30/10/1997

input data	*files*	*results*
V=280 V *h=0.4 µs* *Tc= 200 µs* *i_imp= 0.25* *iimp_c= 0.8* *r_s (expected)=0.1224*	*ST_RES.pas* *unit ASMOTOR1*	*usalm_a =0.0313* *u_cf =0.0990* *isalm_c =0.7971* *isalm_a =0.2493* *rs_stim= 0.1237* *error= 1.07%*

Table 6.3.24 - Simulation results for r_S estimation

- **Rotor resistance estimation results :**

The estimation of the rotor resistance is done after the stator resistance evaluation and $r_S = 0.1224$ from *Table 6.3.24* is part of the input data. Two sets of results are given in the following *Tables 6.3.25* and *6.3.26.*

Different sampling time and different levels of current are used, and errors encountered are around *20%. Section 6.3.4* shows simulation results for the an error sensitive test. Results are given for simulations carried out with the expected value of r_R and for simulations done with the estimated value of r_R.

Sim. 812- 30/10/1997

input data	*files*	*results*
V=280 V *h=0.4 µs* *Tc= 426.66 µs* *i_imp= 0.25* *iimp_c= 0.8* *rs= 0.0811* *r_r (expected)=0.1060*	*ROT_RS2.pas* *unit ASLAFERT*	*MaxUsaRef = 0.1692* *i_maxUsaRef= 0.7782* *im_b = -0.2487* *rr_stim= 0.1278* *error= 20.6%*

Table 6.3.25 - Simulation results for r_R estimation

Sim. 820 - 31/10/1997

input data	*files*	*results*
V=280 V *h=0.4 µs* *Tc= 200 µs* *19 000- 22 000 steps* *i_imp= 0.25* *iimp_c= 0.8* *rs= 0.0813* *r_r (expected)=0.1060*	*ROT_RS2.pas* *unit ASLAFERT*	*MaxUsaRef = 0.1698* *i_maxUsaRef= 0.7788* *im_b = -0.2486* *rr_stim= 0.1287* *error= 21.5%*

Table 6.3.26 - Simulation results for r_R estimation

- **Rotor time constant estimation results :**

Results are given in *Tables 6.3.27* and *6.3.28* for two sampling periods. Rotor time constant is estimated with an error between *5.22* and *12.44%*.

Sim. 822 - 31/10/1997

input data	*files*	*results*
V=280 V *h=0.4 µs* *Tc= 426 µs* *4 500- 5 500 steps* *i_imp= 0.25* *iimp_c= 0.8* *T_r (expected)=0.0565*	*ROT_CT2.pas* *unit ASLAFERT*	*MaxUsaRef = -0.0307* *MinUsaRef = -0.0646* *UsaRef67 = -0.0432* *timp_minUsaRef= 2.0265* *timp_UsaRef67= 2.0899* *Tr_stim= 0.0635* *error= 12.44%*

Table 6.3.27 - Simulation results for T_R estimation

Sim. 823 - 31/10/1997

input data	*files*	*results*
V=280 V *h=0.4 μs* *Tc= 200 μs* *9 700- 11 000 steps* *i_imp= 0.25* *iimp_c= 0.8* *T_r (expected)=0.0565*	*ROT_CT2.pas* *unit ASLAFERT*	*MaxUsaRef = -0.0323* *MinUsaRef = -0.0643* *UsaRef67 = -0.0440* *timp_minUsaRef= 2.0274* *timp_UsaRef67= 2.0868* *Tr_stim= 0.0594* *error= 5.22%*

Table 6.3.28 - Simulation results for T_R estimation

6.3.3 Sensitivity to rotor resistance variation - Simulation results

Simulations were performed to show the level of error introduced by the rotor resistance estimate, when the estimation had a *20.6%* error [Ref: *Table 6.3.25*] . The error in estimation is given by comparison with the available data.

The simulation is realised with a Pascal program which implements the control scheme of *Figure 6.2.4*. In the first simulation, a speed step of 0.1 p.u. is applied. Therefore in *Figure 6.2.4*, ω_{rif} goes from 0 to 0.1 p.u. . The axis that rotates with the rotor-flux vector is noted as *dq* and the corresponding currents i_d and i_q are presented in *Figure 6.3.15*. Case I reports a simulation carried out with the expected value of rotor resistance, that is corresponding to the manufacturer's data or laboratory classical testing. Case II denotes simulations carried out for the estimated value of rotor resistance, taken with a 20% error.

During the transient period, a difference is noticed between case I and case II i_d current , however its level at steady-state shows no change. This can be seen in the enlargement given in *Figure 6.3.16*.

The speed response to the reference is shown in *Figure 6.3.17*. Results for the two cases are plotted on the same graph and seen to be perfectly the same.

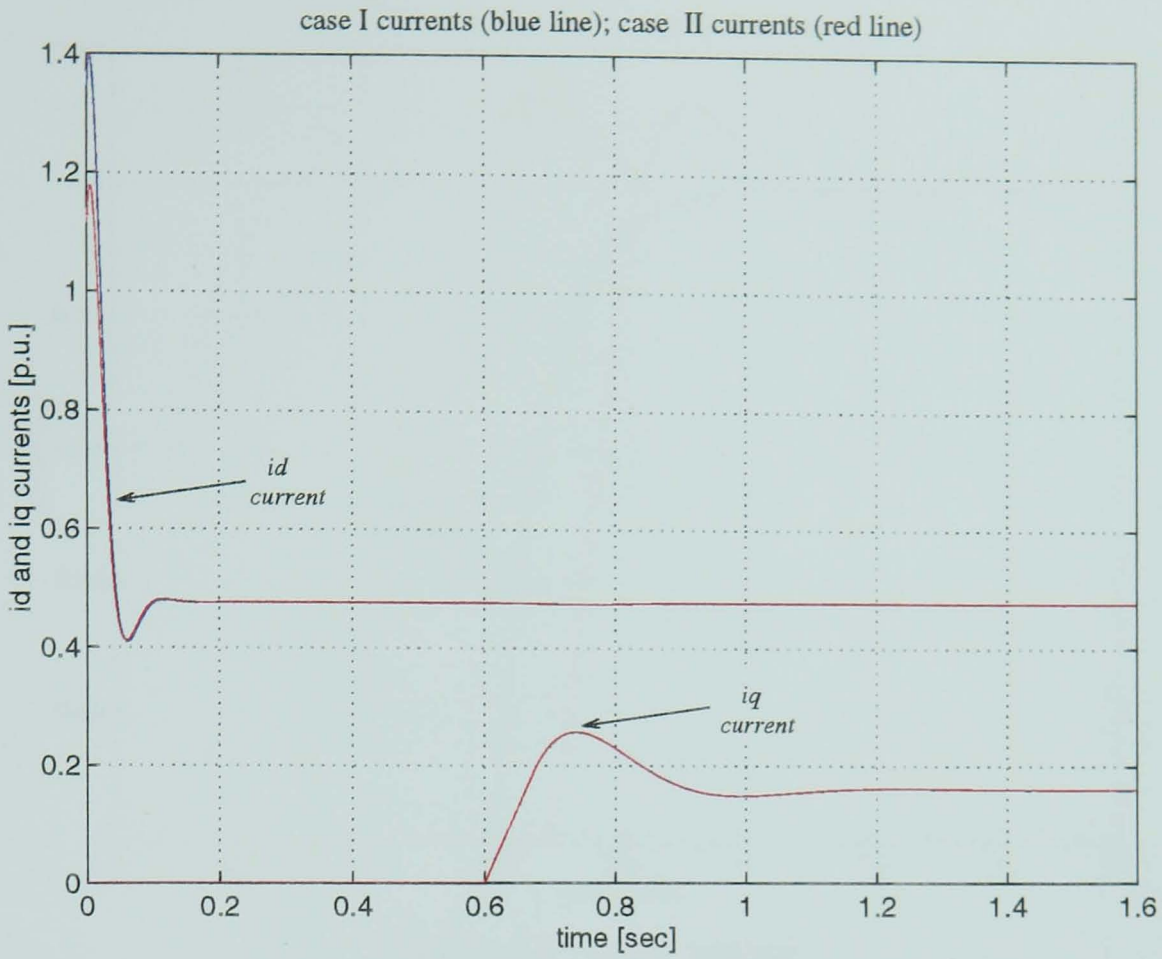

Figure 6.3.15 - id and iq currents

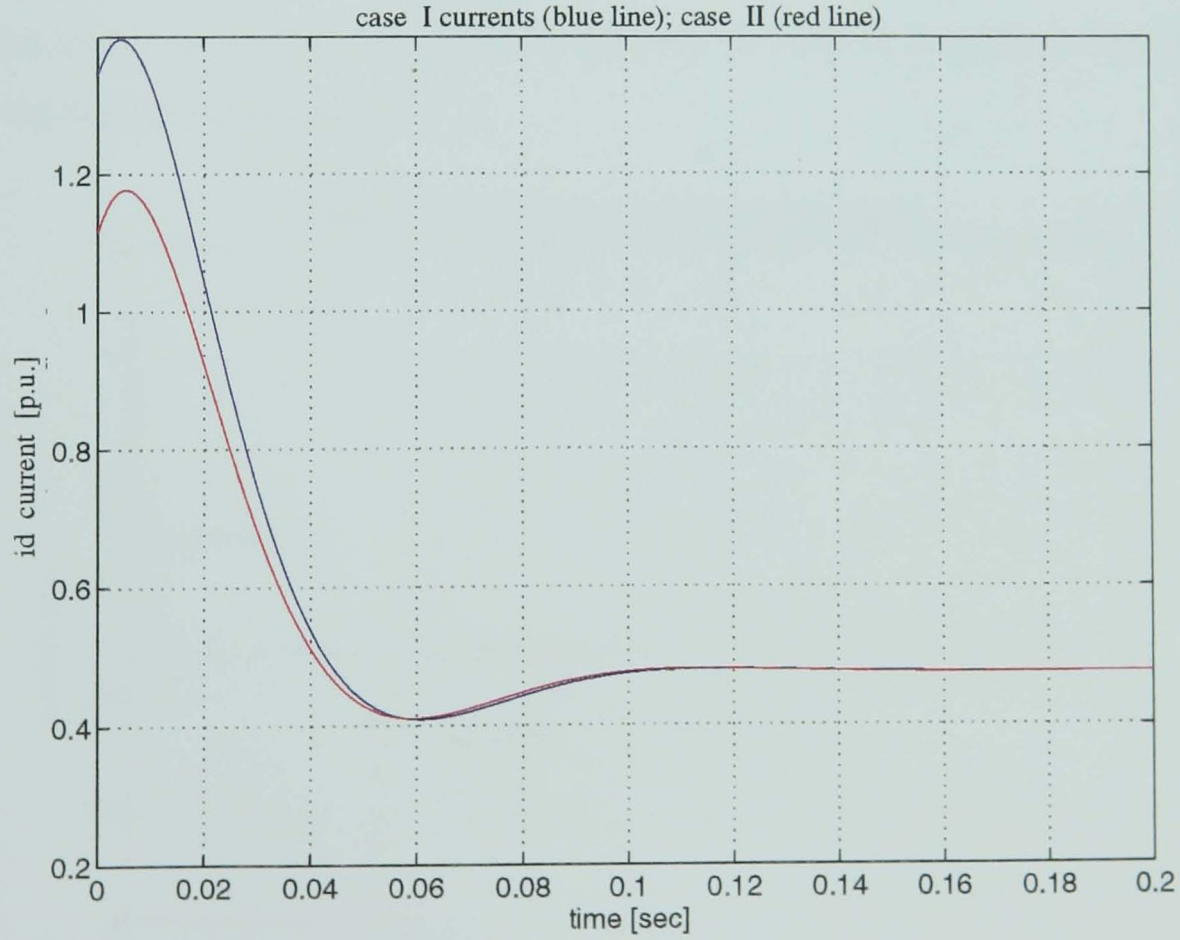

Figure 6.3.16 - enlargement of Figure 6.3.15

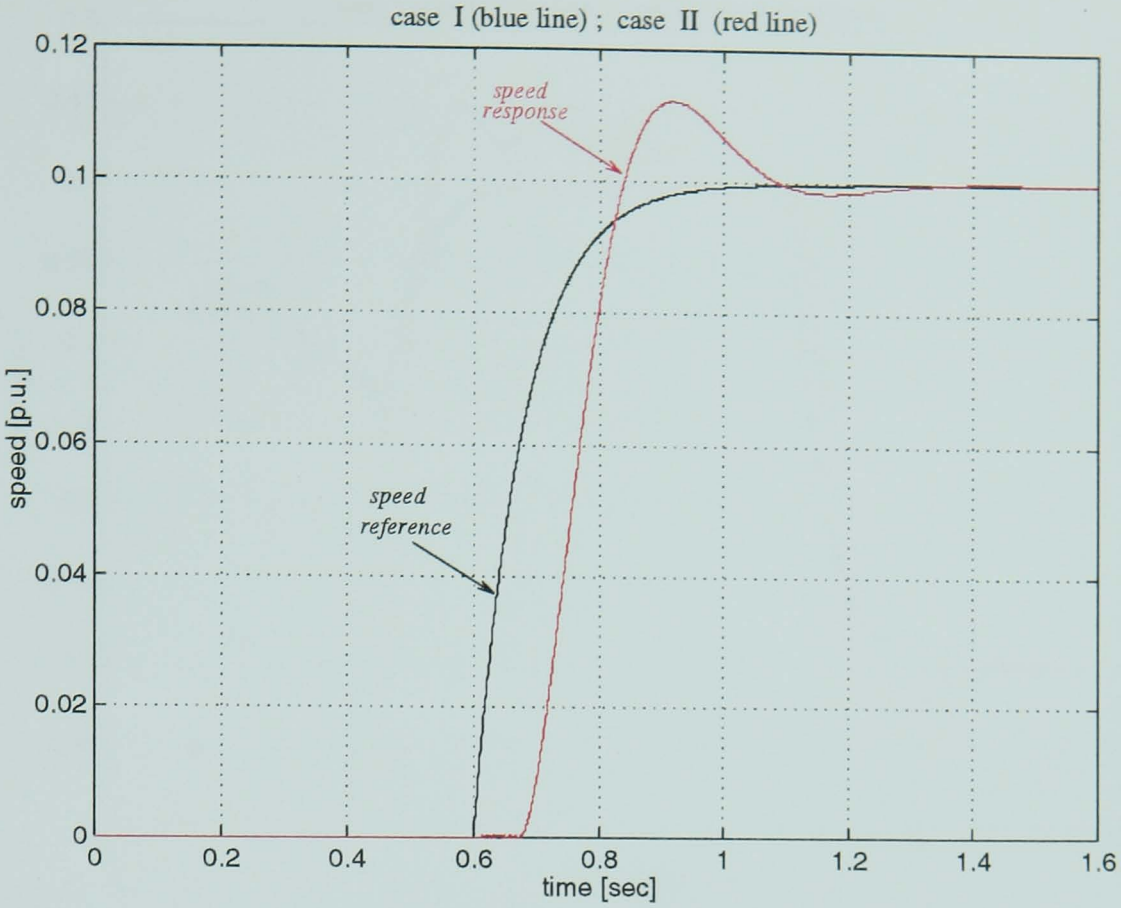

Figure 6.3.17 - speed response

A further simulation was performed to prove the low sensitivity of speed control to rotor resistance changes. A step of *0.4 p.u.* speed was imposed first and after allowing enough time for the motor to reach steady-state, a step back to *0.1 p.u.* is applied. The results are shown in the figures *6.3.18* and *6.3.19*.

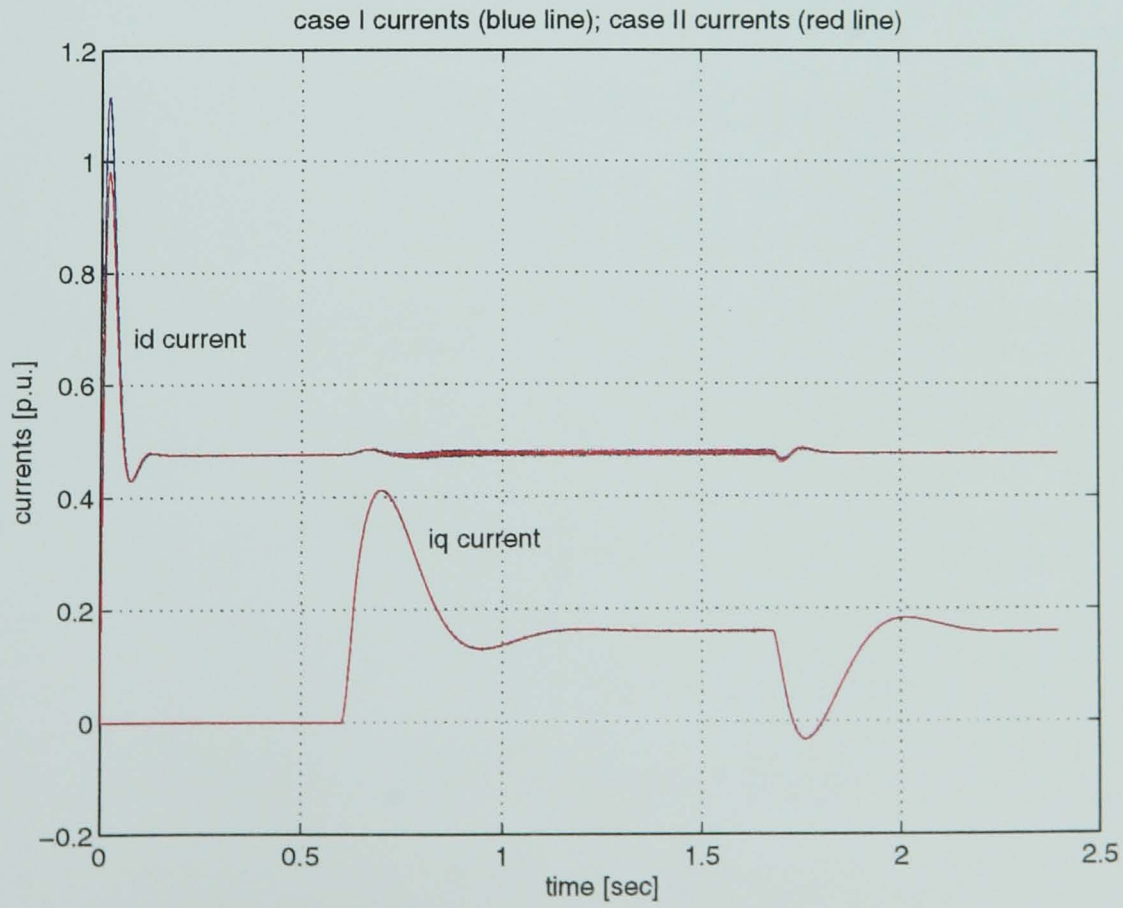

Figure 6.3.18 - current response

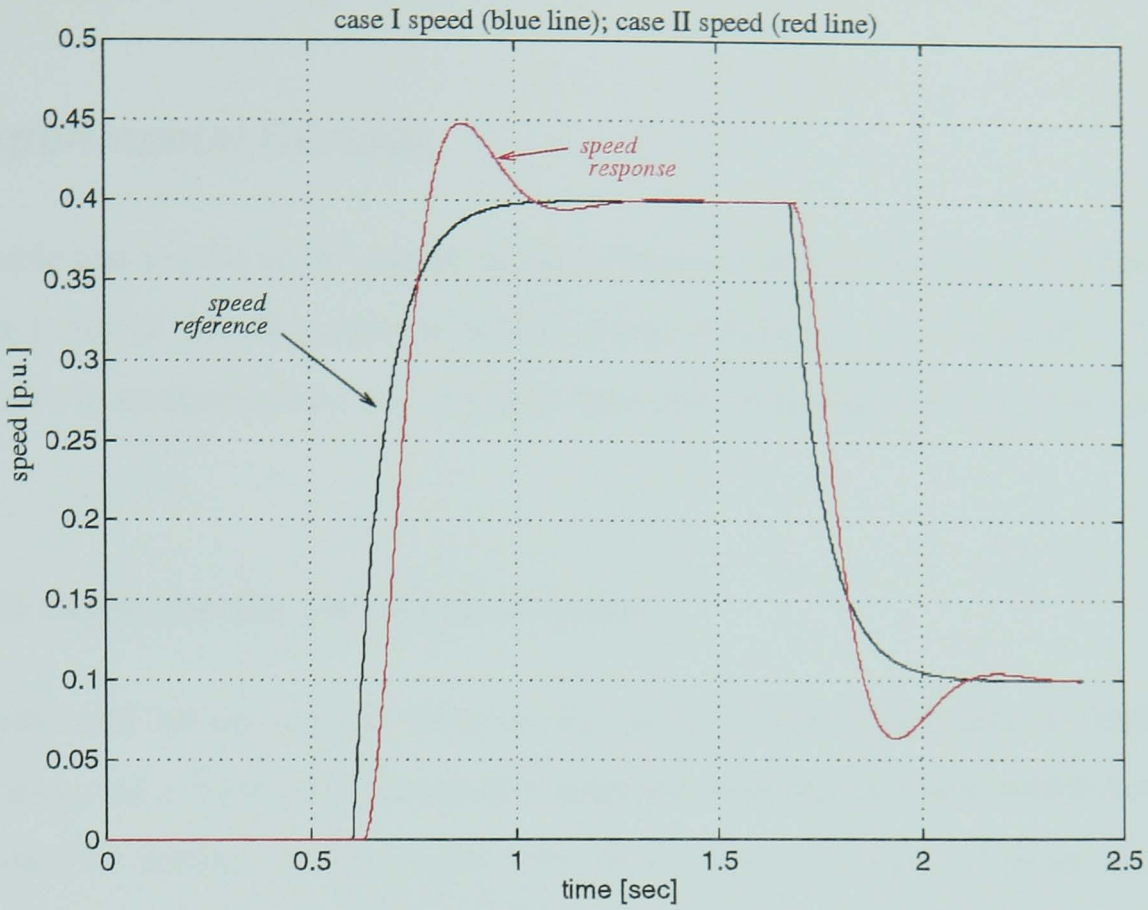

Figure 6.3.19 - speed response

It is seen that the results shown for two simulated cases favour the assumption that there is low sensitivity to rotor resistance variation in the vector scheme, *Figure 6.2.4.*

Following this simulation stage, experimental tests for parameter estimation were carried out. The corresponding results obtained are presented in the next section.

6.4 Experimental Results

To enable test results to be gained an experimental system was used. As this work was carried out at the *Department of Electrical Engineering, University of L'Aquila, Italy*, the test rig available there was utilised. This experimental test-rig is described next.

6.4.1 The experimental set-up description

The experimental set-up can be seen in the picture *Figure 6.4.1* and it consists of an induction motor of *1.5 kW* with an encoder attached to its motor shaft, which however it is not necessary for testing the parameter procedures since the tests are done at stand still. The motor is supplied from a *BJT* voltage source inverter which in turn receives the *d.c.* voltage from a rectifier. The level of *d.c.* voltage can be varied by means of a variac. Further the control algorithm and the estimation procedures are implemented on a *Siemens SAB 80C166* microcontroller. The microcontroller is employed to do all the real time data processing involved in the field-oriented control of the induction motor. The control program and the estimation procedures are both written in assembler language.

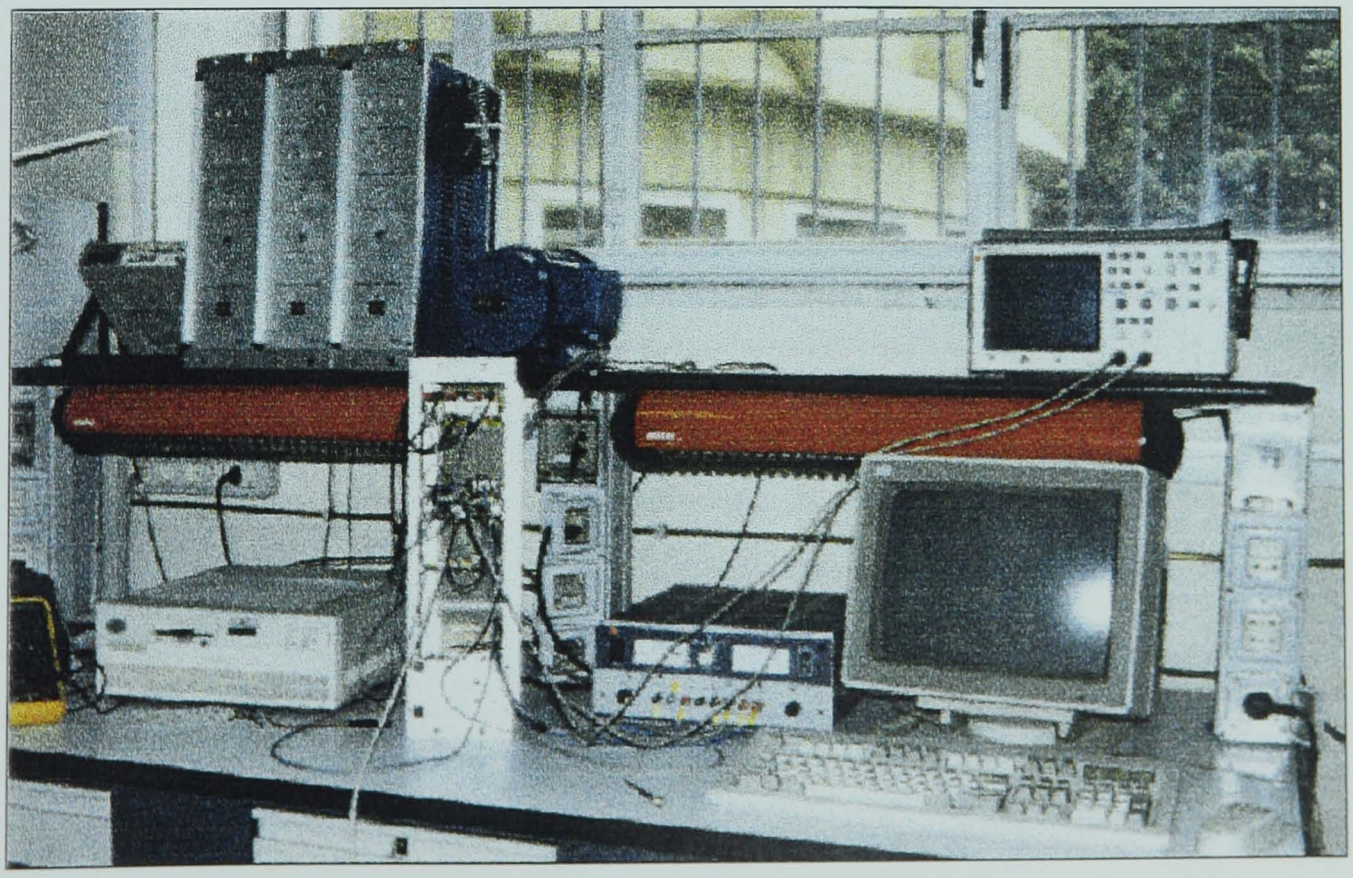

Figure 6.4.1- Experimental set-up at University of L'Aquila, Department of Electrical Engineering

Various interface cards ensure the communication with the host *PC*, the measurement of phase currents or other tasks. The *PC* is dealing with the real-time command and monitoring the drive system. Some other modules are necessary and these were built by the research and technical staff at *Department of Electrical Engineering, University of L'Aquila, Italy.* To enable the measurements of the motor phase currents *Hall* sensors are used and their output is then adapted to the input levels of the *A/D* converter.

A second module filters the encoder output signals and adapts their levels to the input levels of the DMC (*Digital Motor Control*) interface. To perform real-time debugging it is required to be able to see and analyse different quantities on the oscilloscope. This necessitates a module which realises the *D/A* conversion as the variables are in *hexa* format inside the microcontroller memory and need to be transformed to analogue signals to be seen on an oscilloscope.

For communication with the host *PC* a card adapts the serial interface of the DMC board with the *PC* serial port.

The choice of using the *SAB 80C166* microcontroller was an arbitrary one, as at *Department of Electrical Engineering, University of L'Aquila* there were two types of microcontrollers employed in motor drive test-rigs and of those only the *SAB 80C166* was available.

A few characteristics of the *SAB 80C166 microcontroller* are given now: it has a high computational speed, with Instruction Cycle Time of *100 ns* (and pipeline execution of the instructions). It offers 16 registers for general use *GPRs* of 16 bits each with the possibility of being bit addressed and also direct addressed bit manipulation techniques. For tackling exceptions and error conditions hardware traps are provided. It has 1 Kbyte of *RAM* memory. Also an *Interrupt System* is available, *Capture/Compare Unit* and an *A/D Converter* of 10 bits.

Two *Serial Channels* provide connection with other microprocessors, terminals or external peripheral components. It is also equipped with *Parallel ports* and *76 I/O* lines .

The test set-up is now known and consequently test were carried out and results are presented in the subsequent paragraphs.

6.4.2 Results Presentation

- **Total leakage reactance test results :**

Practical tests have employed a motor of *1.5 kW* and its data are presented in Appendix 4 [motor 1 '*LAFERT*']. Sets of results follow, where variables used have changed their names.

From simulation results in *Table 6.4.1* a first set of results is given. The notation are: *Tc=200* μs is the sampling time, $cvd=(1/3)\cdot u_{dc}$ is the voltage pulse applied at the beginning of the *a* interval, and $cvd=-(1/3)\cdot u_{dc}$ is the voltage pulse applied when *c* interval starts.

Resulting variables are: *i3* is the current at *temp3* , the moment when the negative pulse is applied, *i4* is the current corresponding to *temp4* , when the motor is short-circuited and the tests ends. While x_σ indicates the expected value of the total leakage reactance , *x_sig* is the estimated value (in p.u.). The variables used by the estimation algorithm are actually found in *hexa value* in the microprocessor memory, at the specified *addresses*.

16.10.1997 - results:

input data	*variable name*	*address*	*hexa value*	*decimal value*
$Tc= 200\ \mu s$	*i3*	*11B2 h*	*1E2 h*	*0.1177*
$cvd=(1/3)\cdot u_{dc}$	*i4*	*11B4 h*	*F286 h*	*-0.8423*
$cvd=-(1/3)\cdot u_{dc}$	*temp3*	*11B6 h*	*2E h*	*46*
	temp4	*11B8 h*	*3D h*	*61*
expected value of x_σ :				
$x_\sigma=0.213$	*x_sig*	*11B0 h*	*4B4 h*	*0.2939*

Table 6.4.1- Experimental results for x_σ *estimation*

During one test the voltage pulses and the current response have been triggered on the oscilloscope. The graphs so obtained can be seen in *Figure 6.4.2.* and can be compared to those in *Figure 6.3.9* and *Figure 6.3.10.*

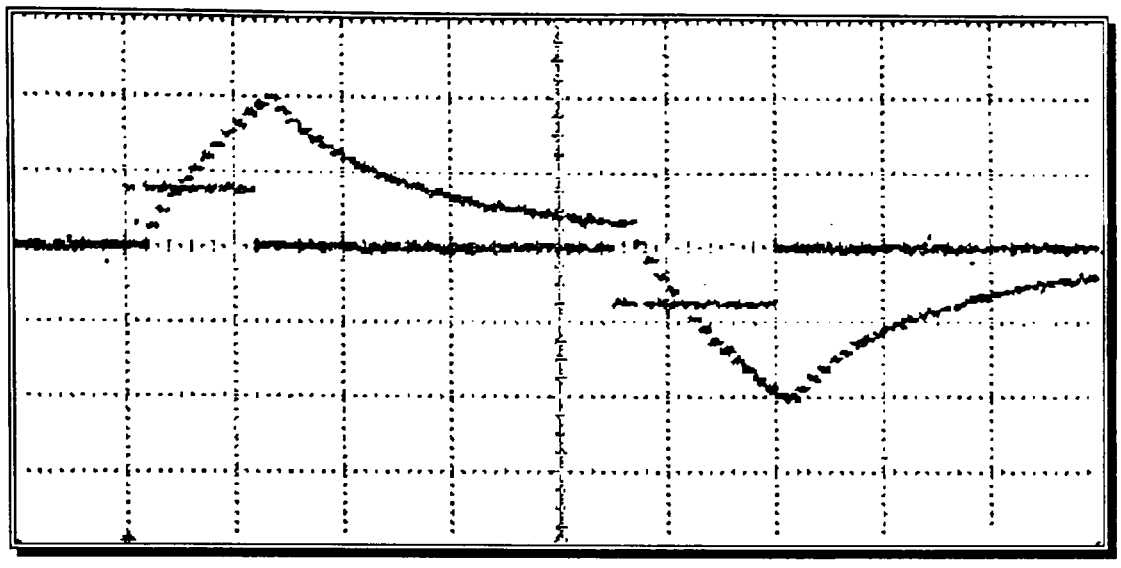

Figure 6.4.2 -The voltage pulses and the current response (on the oscilloscope)

16.10.1998 - results:

input data	***variable name***	***decimal value***
$Tc= 200\ \mu s$	*i3*	*0.0908*
$cvd=(2/3)\cdot u_{dc}$	*i4*	*-0.8320*
$cvd=-(2/3)\cdot u_{dc}$	*temp4-temp3*	*6*
expected value of x_σ *:*		
$x_\sigma=0.213$	*x_sig*	*0.2429*

Table 6.4.2- Experimental results for x_σ estimation

Results for a sampling step of $200\,\mu s$ are presented in *Table 6.4.1* and *Table 6.4.2* and *Table 6.4.3* shows results for $426.66\,\mu s$. The best estimation , *x_sig= 0.2429* is obtained by an error of *14%*.

16.10.1998 - results:

input data:	***variable name***	***decimal value***
$Tc= 426.66\ \mu s$	*i3*	*0.1704*
$cvd=-(1/3)\cdot u_{dc}$	*i4*	*-0.8438*
$cvd=(1/3)\cdot u_{dc}$	*temp4-temp3*	*8*
expected value of x_σ *:*		
$x_\sigma=0.213$	*x_sig*	*0.3176*

Table 6.4.3- Experimental results for x_σ estimation

- **Stator resistance test results :**

Stator resistance estimation has been reported in *Section 6.2.3*. The following three tables show results for practical stator resistance estimation.

The variables are: *i_a* is the current applied on *(a)* interval, *i_b* is the current on *(b)* interval, *i_c* is the current on *(c)* interval, while *u_cfin* is the voltage at the end of *(c)* interval, *usalm_a* is the average voltage on *(a)*, *isalm_a* is average current on *(a)* and similarly *isalm_c* is on *(c)*. The estimated values of the stator resistance are denoted by *r_s* and errors are recorded between *9 - 21%*.

14.10.1997 - results

input data	*variable name*	*hexa value*	*decimal value*
Tc= 426.66 μs	*u_cfin*	*169 h*	
i_a= 0.25 pu	*usalm_a*	*90 h*	
i_b=-0.25 pu	*isalm_a*	*40D h*	
	isalm_c	*CDB h*	
i_c= 0.8 pu			
expected value of r_S			
r_S *=0.12237*	*r_s*	*18A h*	*0.0962 p.u.*

Table 6.4.4- Experimental results for r_S *estimation*

For a sampling step of $200\,\mu s$, even when using different levels of current on the first two intervals, the results are the same. (*Table 6.4.5* and *Table 6.4.6*). *Figure 6.4.3* shows in the top part the current pulses triggered on the scope and correspondingly , the voltage response in the lower part of the plot.

14.10.1997 - results

input data	*variable name*	*decimal value*
Tc= 200 μs	*u_cfin*	*0.0964 V*
i_a= 0.25 pu	*usalm_a*	*0.0356 V*
i_b=-0.25 pu	*isalm_a*	*0.2515 A*
i_c= 0.8 pu	*isalm_c*	*0.8015 A*
expected value of r_S *:*		
r_S *=0.12237*	*r_s*	*0.1104 p.u.*

Table 6.4.5- Experimental results for r_S *estimation*

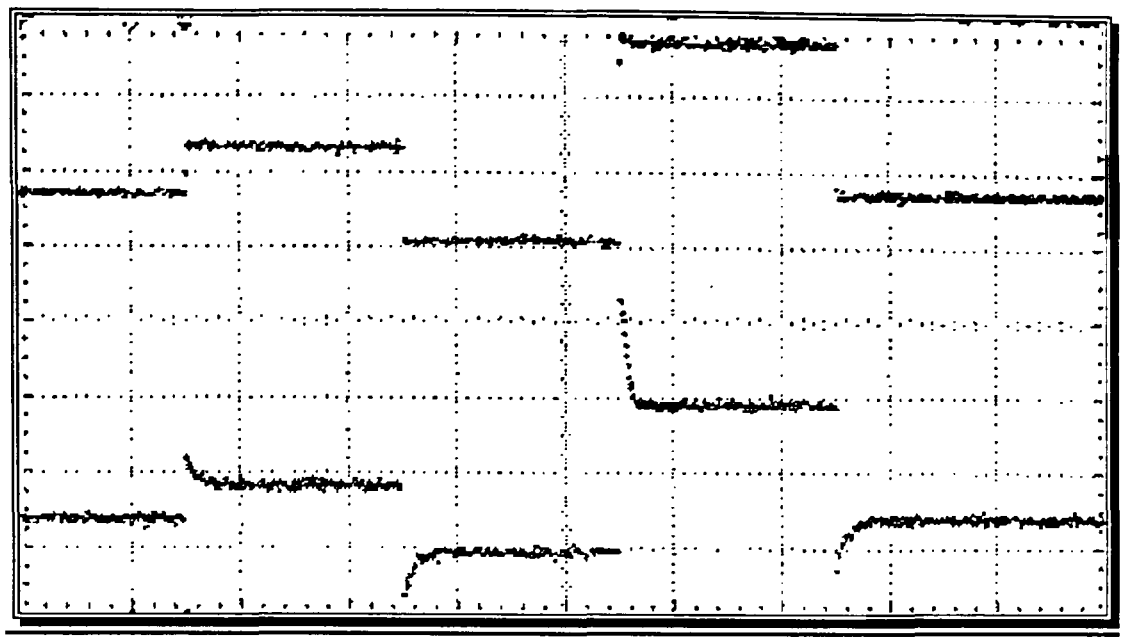

Figure 6.4.3 -The current pulses and the voltage response (on the oscilloscope)

15.10.1997 - results

input data	***variable name***	***hexa value***	***decimal value***
Tc= 200 µs	*u_cfin*	*184 h*	
i_a= 0.15 pu	*usalm_a*	*5Eh*	
i_b=-0.15 pu	*isalm_a*	*26C h*	
i_c= 0.8 pu	*isalm_c*	*CD3 h*	
expected value of r_S:			
r_S =0.12237	*r_s*	*1C4h*	*0.1104 p.u.*

Table 6.4.6- Experimental results for r_S estimation

- **Rotor resistance test results :**

Rotor resistance estimation uses the previously estimated stator resistance. Therefore *r_s* with the lowest error is employed in further calculations. In *Table 6.4.7, isalm_b* is the average current on *b* interval, *u_max* is maximum value of voltage at the beginning of *(c)* interval and *i_umax* , its corresponding current. The number of sampling steps necessary for each interval are denoted by *tau_a*, *tau_b* and *tau_c*. The estimated rotor resistance , in p.u. , is called *r_r*.

20.10.1997 - results

input data	*variable name*	*decimal value*
Tc= 200 μs	*isalm_b*	*-0.2534*
i_a= 0.25 pu	*u_max*	*0.3906*
i_b=-0.25 p.u.	*i_umax*	*0.8433*
i_c=0.8 p.u.		
i_d=0		
rs_stim=1C4 h		
expected value of r_R :		
r_R =0.10595	*r_r*	*0.2715*

Table 6.4.7- Experimental results for r_R estimation

20.10.1997 - results

input data	*variable name*	*decimal value*
Tc= 426.66 μs	*isalm_b*	*-0.1531*
i_a= 0.15 pu	*u_max*	*0.2969*
i_b=-0.15 pu	*i_umax*	*0.8433*
i_c= 0.8 pu		
rs_stim=1C4 h		
expected value of r_R :		
r_R =0.10595	*r_r*	*0.2046*

Table 6.4.8- Experimental results for r_R estimation

20.10.1997 - results

input data	*variable name*	*decimal value*
Tc= 426.66 μs	*isalm_b*	*-0.0532*
i_a= 0.05 pu	*u_max*	*0.2266*
i_b=-0.05 pu	*i_umax*	*0.8433*
i_c= 0.8 pu		
rs_stim=1C4 h		
expected value of r_R :		
r_R =0.10595	*r_r*	*0.1489*

Table 6.4.9- Experimental results for r_R estimation

Results are shown for different levels of motor applied current , however the results are unsatisfactory, showing a minimum error of about *40%* (in the case of *Table 6.4.9).*

It is noted that the error calculation is based on the motor parameter set supplied by the manufacturer and also some parameters may have been out by classical laboratory tests

performed to the motor. Therefore the level of error is relative and in some cases can be taken as satisfactory. Having said this, an error of about 40% is not acceptable and it is thought to be due to error in acquiring the right values of the needed variables at the right moments in time.

This chapter has presented motor parameter estimation. Procedures were developed and simulated in TurboPascal language. By applying voltage and current pulses to the motor at standstill it is possible to calculate rotor and stator resistances, total leakage reactance and rotor time constant. Simulation results have been given for three sets of motor data thus enabling comparison. Errors in evaluating the rotor time constant were encountered for the first two motors and a correcting method has been proposed. In the third motor case, estimation of the rotor resistance has been achieved by an error around 20%. Therefore it was thought appropriate to run sensitivity tests utilising the erroneous value of rotor resistance. Speed steps from standstill to *0.1 p.u.* or to *0.4 p.u.* were applied and results shown prove a very low sensitivity of the vector control scheme to the rotor resistance variation.

Chapter 7

Implementation of rotor flux oriented control

Pulsewidth-modulated (*PWM*) inverters, used in conjunction with cage induction motor (*CIM*), are now a recognised a.c drive technology chosen on the basis of both performance and cost. In most industrial situations the advantage of the drive is the *CIM* itself, which is rugged, low-cost and reliable, requiring little maintenance for a long working life. Market studies prove the general applicability of a.c. induction motors in industrial applications. However the use of the induction motor has disadvantages in that it is difficult to control.

Traditionally, motor control was implemented with analogue components, however nowadays this type of control is almost entirely replaced by digital control. When DSPs (*Digital Signal Processors*) are employed further improvements are possible which lead to higher speed, higher resolution drives and also sensorless control supported by control algorithms.

Scalar control has drawbacks, for example poor reaction time to load changes, consequently vector control has succeeded in being adopted as standard for controlling the induction motor in adjustable speed applications with changing load and speed references.

The objective of this chapter is to present a digital implementation of a rotor flux control strategy for the inverter-fed induction motor previously modelled.

7.1 Hardware and system elements description

A vector control scheme is presented in *Figure 7.1.1* and this is implemented using the experimental set-up which is schematised in *Figure 7.1.2*.

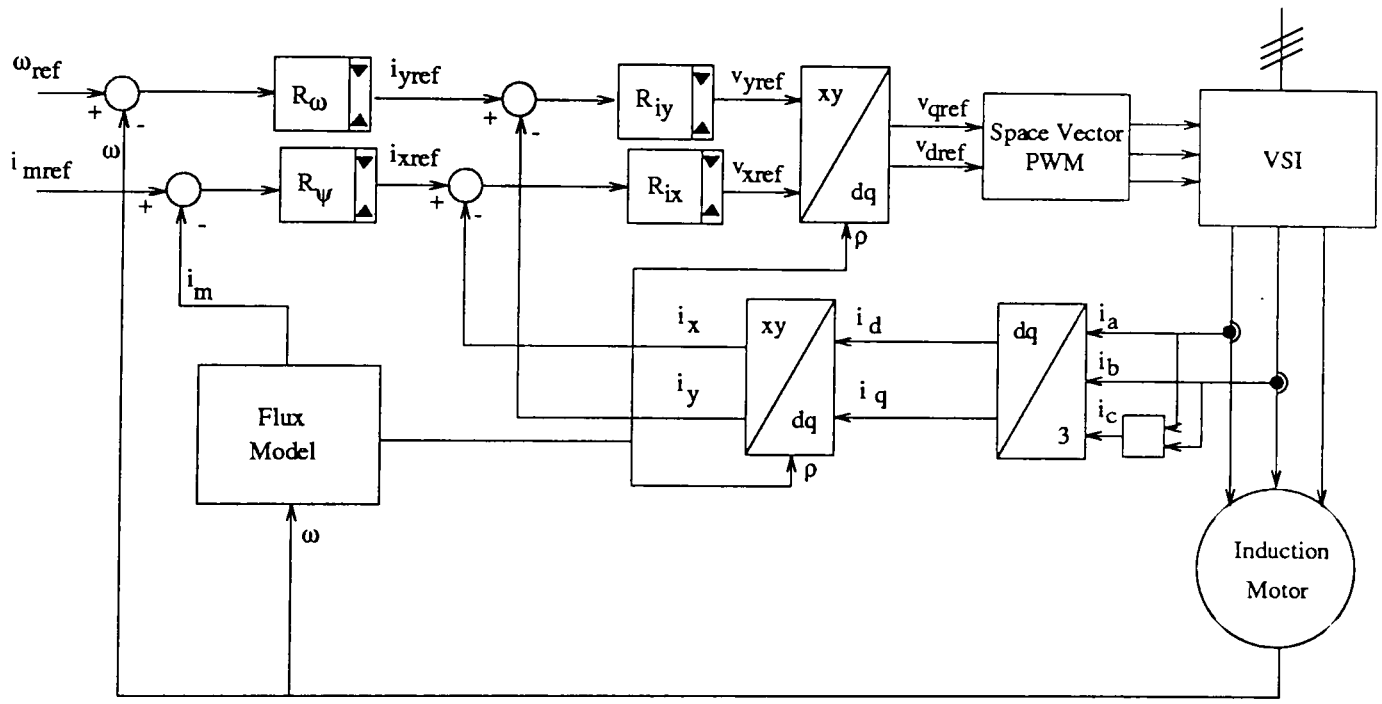

Figure 7.1.1 - Vector control scheme

The control algorithm, processing of external data, *PWM* modulation and the derived switching sequence for the *IGBT* inverter is realised by a DMC (*Digital Motor Control*) board. This is the *TMS320F240*, manufactured by *Texas Instruments Inc.*

As the performance of an a.c. drive is dependent on the effectiveness of its control, the use of a DSP controller offers advanced opportunities, for example the possibility of implementing a real-time algorithm and also sensorless control. The *TMS320F240* device chosen contains a *16-bit* fixed-point DSP core with microcontroller peripherals, such as memory, PWM generator, A/D converters, and is therefore ideal for motor control strategies development. Fixed-point DSPs are favoured for motor control, as they are

cheaper than the floating-point DSPs and for most applications a range of 16 bits is enough.

The TMS320F240 EVM (*Evaluation Module*) is equipped with an event manager that provides all pulse-width modulation (PWM) and I/O features to drive all motor types, an *SPI* and *SCI* interfaces for serial communications, 28 bi-directional I/O pins, a watchdog timer and integrated dual 10-bit *A/D* converters.

The motor phase currents are measured by *Hall*-effect based sensors and their voltage output is sent to the A/D (*Analog/Digital*) converter on the *EVM* board. However as the *Hall* sensors output may be considered in the range -2.5 ÷ +2.5 V (-4÷ +4V for full range) and the *A/D* converter can only digitise voltages from 0 ÷5V, an auxiliary circuit was built. The circuit is based on *Operational amplifiers* and a *Voltage Reference* source and it realises the offset of the *Hall* sensors voltage output to the range 0 ÷5V.

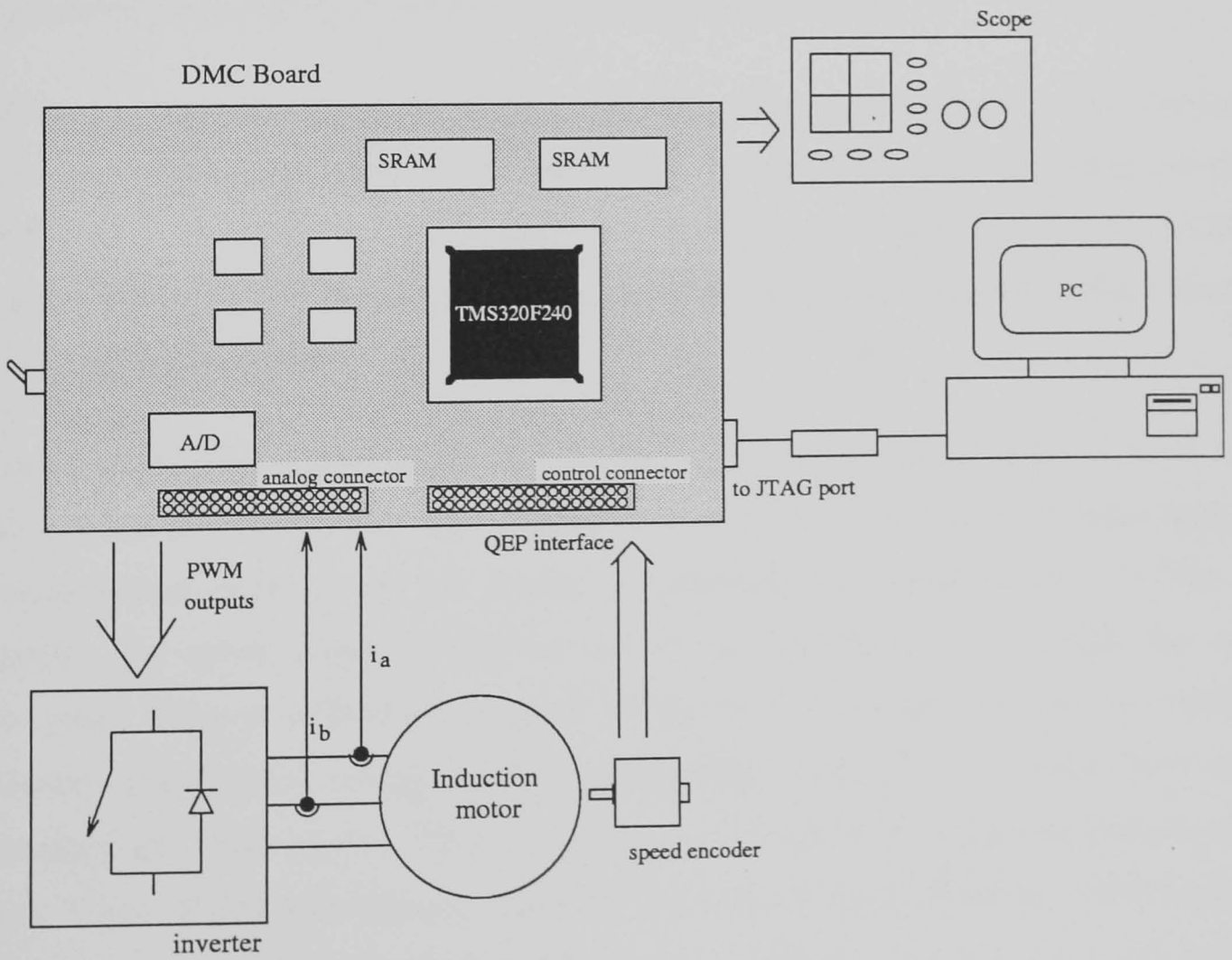

Figure 7.1.2 - Experimental set-up

Consequently, motor currents are measured and are taken as external input data for the control scheme.

In addition a speed estimation encoder is attached to the induction motor shaft. It is well known that most researchers favour sensorless control, however it was desired to realise this scheme with speed measurements as a starting point and then to introduce by further development a speed estimator. A *Kalman* filter algorithm for example, may be employed for this task.

As a result of the control algorithm based on current and speed measurements and an external speed reference, which for experimental purposes is given from the PC keyboard, six PWM pulses are sent by the board to the inverter. As these pulses have *5V* in amplitude they will go first to a boost circuit which brings them to a *15V* level thus permitting them to drive the IGBTs.

The control scheme in fact generates three PWM outputs and another three outputs are provided automatically and implicitly they represent the negations of the original three signals. A dead band can also be programmed to values from 0.05 μs to 3.2 μs by programming a specific register, DBTCON of the *TMS320F240* .

Editing the assembler program, linking and debugging is realised on a *PC*. Through a *JTAG* scan path connected to the *PC* through the parallel port and on the other end to the *JTAG* port on the *DMC* board, the actual download of the program is realised. The *JTAG* scan permits an on-line debug of the program by inspecting the registers and any sections of the memory as needed.

A serial connection is also established between the microcontroller board *RS-232* serial port and the PC for at least two reasons. One is to download memory data from the microprocessor memory and the second to command the microprocessor on-line. For example, the speed command can be sent to the microprocessor through this serial connection using as software a program written in C language. The control algorithm computes the desired voltage at the corresponding frequency and these have to be modulated as PWM signals. This is realised by a modulation technique known as the Space Vector PWM. Consequently the DSP board sends the appropriate PWM signals to the inverter and the motor is then supplied with a voltage V and frequency f in order to achieve the speed reference.

The next section introduces the *Space Vector Modulation* used in generating the PWM signals required by the inverter.

7.2 The Space Vector PWM switching technique

Traditionally, the PWM method for a.c drives was implemented using analogue technique. However with simple analogue electronics it is not possible to produce three exactly balanced analogue signals over the required voltage and frequency range.

Therefore the control of a.c. drives has moved towards an exclusively digital implementation. It is possible to have a digital equivalent of the comparison between a sawtooth carrier and a modulating sinewave. The technique is known as *regularly sampled PWM* because the PWM signals are generated by comparison of a regularly sampled version of the modulating function with a carrier. The analogue scheme is in contrast called *naturally sampled PWM* [85] .

Recently, a new modulation technique has been evolving, known as the SVPWM, the *Space Vector Pulse Width Modulation*. Beside being a technique which is perfect for digital implementation, it offers a better utilisation of the *d.c.* link voltage and a lower harmonic content, particularly at high modulation indices.

The Space Vector PWM technique is defined by the approximation of V_{ref}, the reference voltage, which can be defined as:

$$V_{ref} = v_{dref} + jv_{qref},$$

using a combination of the eight switching patterns. Depending on which sector contains V_{ref} , the two vectors (*V1* and *V2* for *sector 1*, for example) are applied for t_i and t_j, respectively.

For the vector control scheme, previously shown in *Figure 7.1.1* , V_{ref} is given by v_{dref} and v_{qref}. As the desired output is a set of three phase sinusoidal voltages, V_{ref} is a vector rotating around the origin of the *DQ*-axis plane. Its frequency is given by that of the desired three phase voltages.

On the inverter side, six command signals are needed to drive the IGBTs. These can be the command signals: *d0*, *d1*, *d2* and *d3*, *d4*, *d5* which are the respective complementary

signals of the former. They can be seen in *Figure 7.2.1* where a typical structure of a three-phase power inverter is shown.

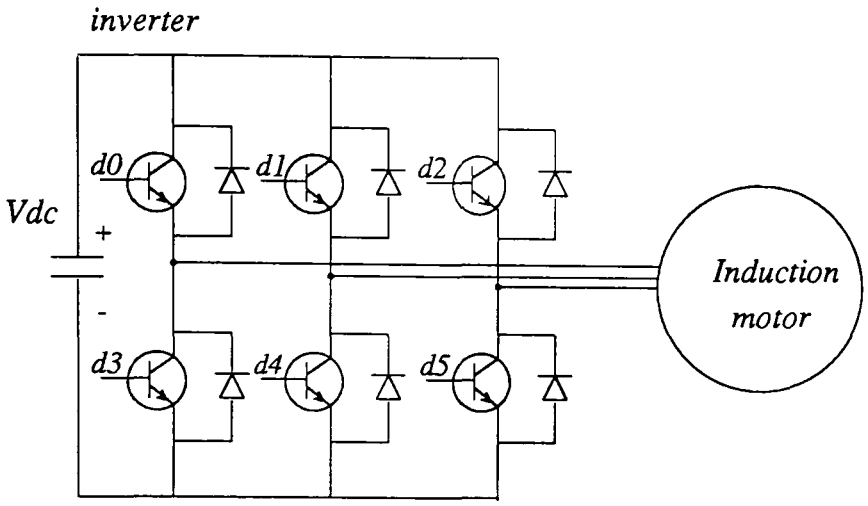

Figure 7.2.1 - Typical scheme of a three-phase power inverter

For a.c. motor control when one upper transistor is switched on, the corresponding lower transistor is switched off. Therefore the states of the upper transistors are sufficient to evaluate the output voltages: v_a, v_b and v_c, applied to the motor windings. There are eight possible combinations shown in *Table 7.2.1*.

d0	*d1*	*d2*
1	0	0
1	1	0
0	1	0
0	1	1
0	0	1
1	0	1
0	0	0
1	1	1

Table 7.2.1

Out of the eight possible states, one case is taken and shown in *Figure 7.2.2*.

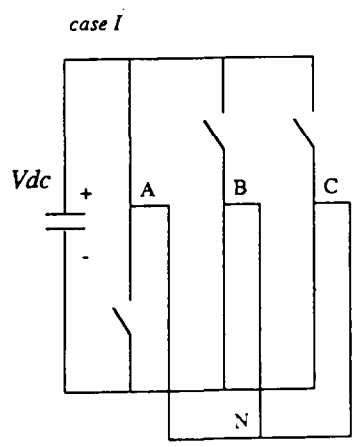

Figure 7.2.2 -case I of the inverter switches

The voltages applied to the motor can easily be evaluated for this case as:

$$v_a = V_{AN} = \frac{2}{3}V_{dc} \text{ and } v_b = v_c = V_{BN} = V_{CN} = -\frac{1}{3}V_{dc} \qquad (7.2.1)$$

where V_{dc} is the d.c. link voltage. The voltage space vector in *case I* is of length $\frac{2}{3}V_{dc}$ and lies on the real axis , that is the *d*-axis as can be seen in *Figure 7.2.3*.

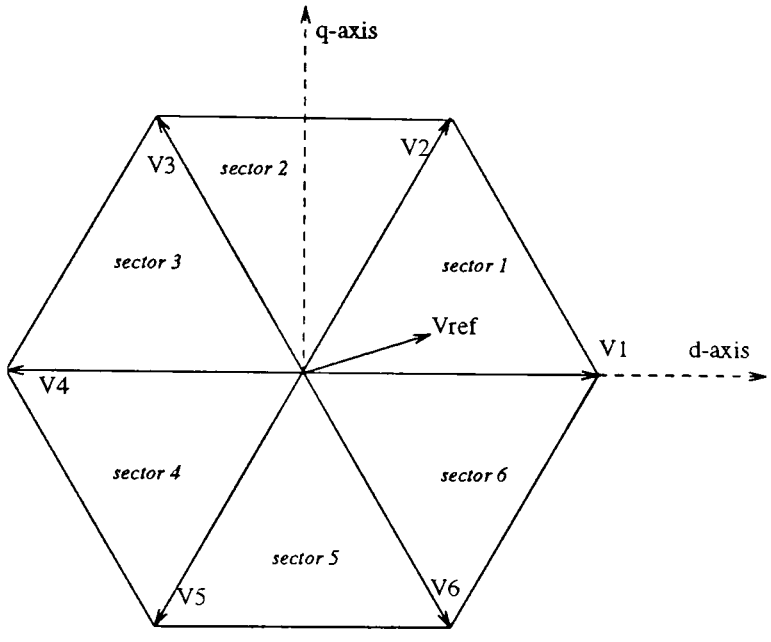

Figure 7.2.3 - The phase voltages in DQ - axis plane

For the other states a similar calculation can be performed and the phase voltage corresponding to these eight combinations can be represented in the *DQ*-axis plane by the hexagonal axes in *Figure 7.2.3*. From *Table 7.2.1* there are six non-zero vectors forming the hexagon axes and two zero vectors which are at the origin.

Therefore using the *Space Vector* modulation technique any voltage space vector belonging to this hexagon can be realised.

7.2.1 Modulation index

If a system of three-phase balanced voltages is defined as:

$$\begin{aligned} V_{AN} &= V\cos(\omega t) \\ V_{BN} &= V\cos(\omega t + \frac{2\pi}{3}) \\ V_{CN} &= V\cos(\omega t + \frac{4\pi}{3}) \end{aligned} \qquad (7.2.2)$$

Then the corresponding space vector is:

$$V_{sv} = V[\cos(\omega t) - j\sin(\omega t)] = Ve^{-j\omega t} = M(\frac{V_{dc}}{2})e^{-j\omega t} \qquad (7.2.3)$$

where M is called the modulation index, given by:

$$M = \frac{V}{(\frac{V_{dc}}{2})} \qquad (7.2.4)$$

where V is the peak fundamental magnitude and Vdc is the d.c voltage.

The maximum fundamental phase voltage is $V_{\max} = \frac{V_{dc}}{\sqrt{3}}$ and thus the modulation index becomes:

$$M = \frac{\frac{V_{dc}}{\sqrt{3}}}{(\frac{V_{dc}}{2})} = \frac{2}{\sqrt{3}} = 1.15.$$

Compared to the maximum modulation index of *1.27* obtained for an inverter with a given d.c. link under *six-step operation*, the M_{max} for *SVPWM* offers a *90.69%* utilisation of the inverter capability. This also exhibits an increase of *15%* over the conventional sinusoidal modulation where M_{max}=1.

When the vector control is implemented together with the SVPWM technique, the outputs of the current controllers are v_{qref} and v_{dref} and they will determine one of the six possible sectors, to which the space vector belongs.

Although it is not necessary to use the adjacent inverter states in the synthesis of the output voltage, it can be shown that superior harmonic performance is obtained when this condition is satisfied [85].

7.2.2 Estimation of the active state times

A single PWM cycle is given by the sequence:

$$u_0 \rightarrow u_a \rightarrow u_b \rightarrow u_7 \rightarrow u_7 \rightarrow u_b \rightarrow u_a \rightarrow u_0$$

where: u_0 *and* u_7 are the two null voltage vectors and

u_a *and* u_b represent the two inverter output vectors immediately adjacent to u_s (V_1 *and* V_2 for example , in *sector 1*).

Equating the volt-second integral of vector V_{ref} with the inverter output voltage vectors over a single space vector modulation cycle gives:

$$V_{ref}T_{PWM} = u_0 t_0 + u_a t_i + u_b t_j + u_7 t_7 \qquad (7.2.5)$$

where $T_{PWM} = t_0 + t_i + t_j + t_7$, T_{PWM} is the PWM period,

t_i is active time when u_a is applied,

t_j is active time when u_b is applied and

t_0 *and* t_7 are the periods when u_0 and u_7 are applied.

The time intervals are derived as:

$$t_i = \frac{3}{2} V_{ref} T_{PWM} (\cos\alpha - \frac{1}{\sqrt{3}} \sin\alpha)$$

$$t_j = \sqrt{3} V_{ref} T_{PWM} \sin\alpha \text{ and} \qquad (7.2.6)$$

$$t_0 + t_7 = T_{PWM} - (t_i + t_j) = T_{PWM} - T_{active}$$

where α represents the intrasextant displacement of V_{ref},

T_{active} is the *active* modulation time and

V_{ref} is in p.u. when V_{dc} is defined as *1 p.u.*

Having derived the active periods of time for a particular modulation cycle, it is possible to produce the switching signals to be applied to the inverter IGBTs. There are various sequences in which the inverter states can be applied however if a transition from one state to another is made with only one inverter switch then a minimum inverter switching frequency can be attained.

Therefore a switching pattern similar to *Figure 7.2.4* is obtained in each sector. When this type of sequence $u_0 \rightarrow u_a \rightarrow u_b \rightarrow u_7 \rightarrow u_7 \rightarrow u_b \rightarrow u_a \rightarrow u_0$ is applied the technique obtained is known as *double edge* space vector modulation and is by far the most widespread due to inherent harmonic advantages.

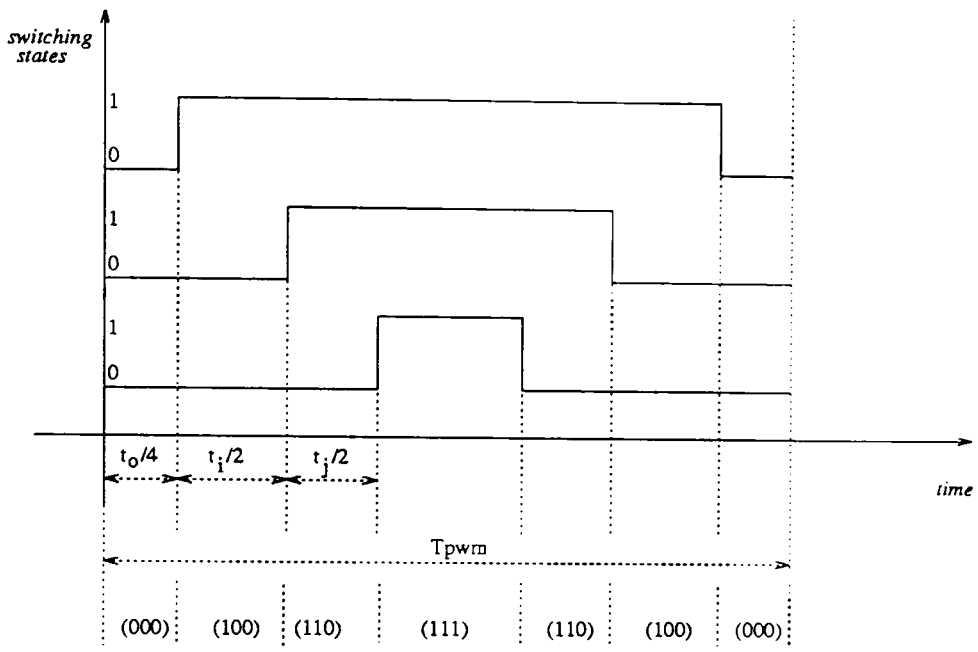

Figure 7.2.4 - An example of space vector PWM switching pattern

The total zero time $t_0 + t_7 = T_{PWM} - (t_i + t_j) = T_{PWM} - T_{active}$ is conveniently divided in equal periods. This is a classical approach. Since the effect of both zero vector states is identical, a degree of freedom can be taken in how each zero state is used. This enables production of alternative space vector modulation pulse sequence and therefore the exploration of the PWM output.

One of these alternative methods is generally known as the *bus clamped* modulation technique [86]. This uses only one zero state and thus an inverter switch is clamped to either the negative or positive d.c. link for a number of modulation cycles while the other two switches are switched using the SV technique.

Limiting the applied voltage vector may be also necessary in various situations. By hexagonal limit it is understood that the voltage vector has to lie along the hexagonal inverter limit [85].

In this case $(t_i + t_j) > T_{PWM}$, for different values of α, requiring scaling of t_i and t_j:

$$t_{i\_new} = \frac{t_i}{t_i + t_j} T_{PWM} \text{ and } t_{j\_new} = \frac{t_j}{t_i + t_j} T_{PWM} = T_{PWM} - t_{i\_new} \qquad (7.2.7)$$

In the subsequent paragraphs, two modalities of implementing the Space Vector Modulation on the TMS320C240 microcontroller are succinctly presented.

7.2.3 Implementing the Space Vector PWM technique on TMS320F240

There are two possibilities when the SVPWM technique is implemented on the TMS320F240 controller. One will produce a *bus-clamped* modulation and is described by the steps :

- configure *COMCON* and *ACTR* registers
- choose one of the timers and set it in continuous-up/down counting mode
- evaluate v_{dref} and v_{qref} and also the sector
- calculate t_i , t_j and t_0
- write the switching pattern in register *ACTR*
- load into the compare registers as follows: $(t_i / 2)$ into *CMPR1* and $(t_j / 2)$ into *CMPR2*.

The second choice gives a *double-edge* modulation. At every modulation cycle the channels to be toggled have to be determined according to the sector to which the reference voltage vector belongs.

- CMPR1, CMPR2 and CMPR3 , the comparing registers are enabled and initialised
- timer T2 set-up as having the period equal to the sampling period of the control loop
- PWM1, PWM3 and PWM5 , three of the 9 available PWM outputs , are configured as being active high, meaning that at first compare match they will toggle from low to high,

- PWM2, PWM4 and PWM6 are set-up as active low as they are the negated signals of PWM1, PWM3 and PWM5 respectively and also corrected by an appropriate dead-band given in DBTCON,

- *ti*, *tj* and *to* are calculated according to the active sector and appropriate values are written to the compare registers , CMPR1 - 3.

7.3 Assembly implemented algorithm

Having described the experimental set-up and the PWM modulation technique, this section presents the software, i.e. the control algorithm implemented as an assembler program.

The program which implements the control scheme and manages all the information is written in assembler language and its flow chart is shown in *Figure 7.3.1*. The assembly language is for the *TMS320F240* digital signal processor from *Texas Instruments*.

To prepare the board for this implementation a few steps have to be followed as part of the initialisation. Firstly, the systems registers are initialised (registers which set the CPU clock, manage the watchdog, the shared I/O ports, incoming interrupts etc.), then the variables to be used by the program are declared for memory purposes. In most cases these will occupy a *word* and most are initialised to zero.

In addition, the timers and the compare unit are set-up. On the *EVM* three general purpose timers are available. Timer *T1* implements the PWM period which may be taken as equal to the sampling period, implemented by timer *T2*. The sampling period, function of the system clock, may be *400* μs for example. Whereas timer *T3* will count the signals coming from the encoder through a special *QEP* (Quadrature Encoder Pulse) circuit. Therefore these timers have to be set-up accordingly.

Other tasks of the initialisation routine are: set-up of mask registers, clearing the flag registers, configuring the *A/D* converter to read two channels simultaneously and starting the conversion. Once all the operations are completed, the timers are started.

It can be seen from the flow-chart that after the initialisation part is finished, the program arrives at a test point. The actual program is started when an interrupt comes from timer *T2* underflow. Thus the sampling time is controlled and is $400\,\mu s$, this value being dependant on the system clock, which has a frequency of *20 MHz* and therefore a period of *50 ns*.

When the sampling period starts the following steps are executed:

- timer *T3* , which is clocking the *QEP* circuit , is read and motor speed evaluated
- the *ADCFIFO* registers are read and the digital results of *ADC* unit are obtained
- by appropriate scaling the motor phase currents are evaluated
- determine the i_d and i_q currents using formula from *(2.5.1)*
- θ_r , the rotor flux angle, is calculated, and then using look-up tables $\sin(\theta_r)$ and $\cos(\theta_r)$ are determined
- transformation of the current from *DQ*-axis to *xy*-axis, the frame fixed to the rotor-flux space vector : i_d and i_q to i_x and i_y by formula *(5.5.27)*
- implement the *Flux PI Regulator* and calculate i_{xref} as its output
- implement the i_x *Current PI Regulator* and calculate v_{xref} as its output
- implement the *Speed PI Regulator*, which gives i_{yref} as its output
- implement the i_y *Current PI Regulator* and v_{yref} as its output
- transformation of v_{xref} and v_{yref} back to *DQ*-axis into v_{dref} and v_{qref}
- transformation of v_{dref} and v_{qref} to the desired three-phase voltages : v_a, v_b and v_c
- evaluation of the sector [*SVPWM*]

- calculation of t_i, t_j and t_o duration of time for which the switching patterns are applied to the inverter (t_o for applying the zero vector, pattern: 000 or 111)

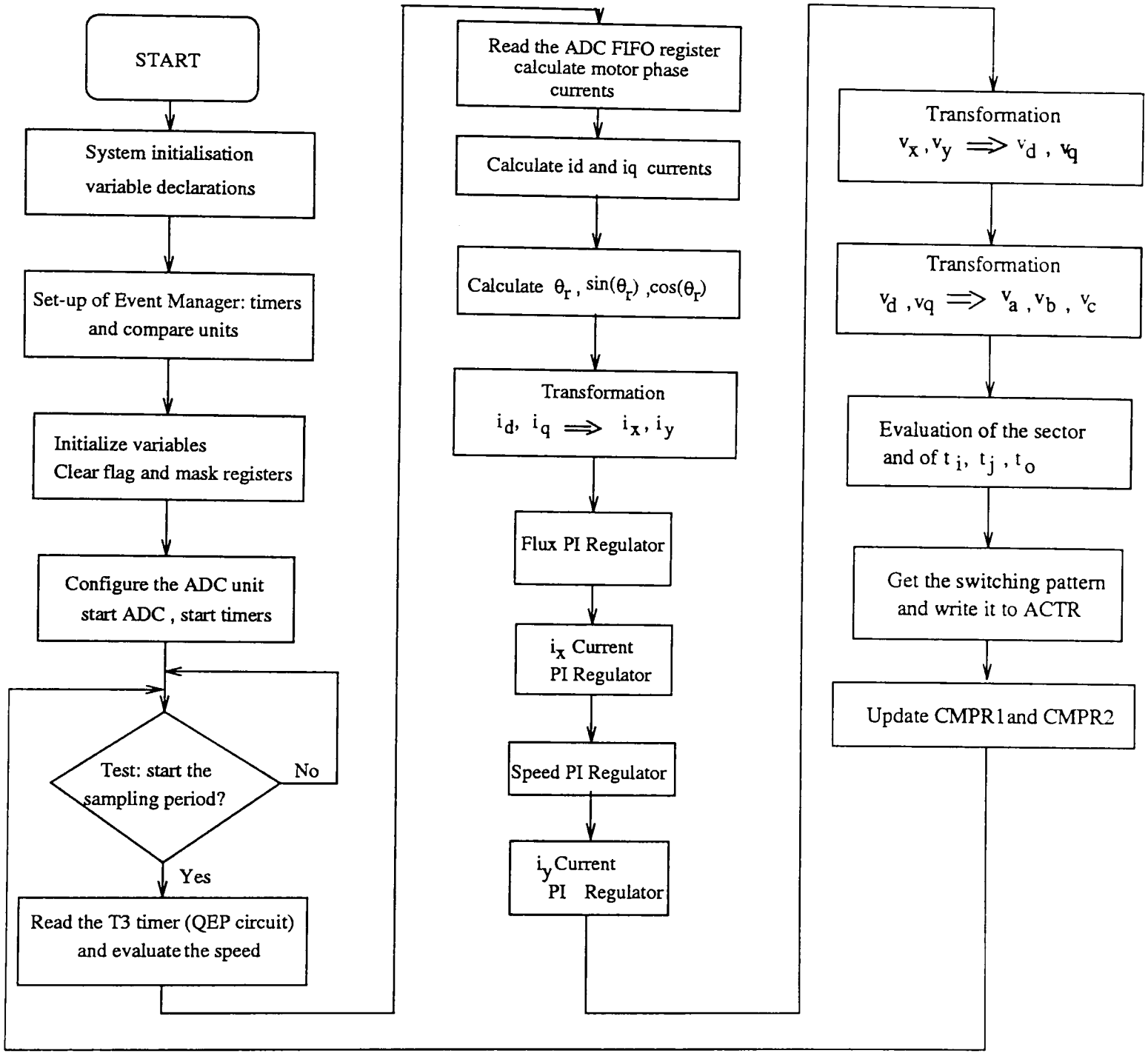

Figure 7.3.1 - Program flow chart

Thus the hardware set-up has been presented and described. Further the control program, shown as a flow-chart, has also been briefly discussed. Application of this experimental set-up to a motor test system has been achieved and tests have been performed. Test results are given in the next paragraphs.

7.4 Experimental Results - Vector Control Implementation

The motor used for tests is a *1.5 kW* induction machine, working at *50 Hz*. with squirrel-cage rotor. When connected in *delta*, the stator windings have a rated current of *5.6 A* at 220- 240 V. The motor has two poles and a rated speed of *2820 rpm*. Tested by classical procedures of locked-rotor and no-load tests, the motor was found to have the following parameters :

the rotor resistance R_R= 4.17 Ω,

the stator resistance R_S= 5.7 Ω,

the leakage reactances: $X_S = X_R$= 6.03 Ω.,

magnetising reactance X_M= 142.92 Ω,

The rotor time constant is : T_R= 0.01137 .

The test-rig and its components can be seen in the pictures of Appendix 5. The listing of the assembler program is given in Appendix 6.

To test the response of the control system a stepwise speed control increment is applied to the drive system. The motor is at standstill and in no load condition when the step is applied. The following figures present test results for the case of a speed step of *0.05 p.u.* being applied to the motor.

The speed in per unit is calculated as $speed[p.u.] = \frac{rotor\ speed[rpm]}{base\_speed[rpm]}$, where the base speed is *3000 rpm*, the synchronous speed. The encoder measured data consist of samples of speed at every 10 sampling steps.

Speed response of the motor is plotted versus time in *Figure 7.4.1*. The data have been first stored in the DSP memory and then transmitted to the host PC through the serial port. Speed acquisition from data recorded by the QEP interface on the DSP board is performed every 10 sampling steps to avoid errors. The time/speed plot shows a good response of the motor to a speed reference of 150 rpm (0.05 p.u.) .

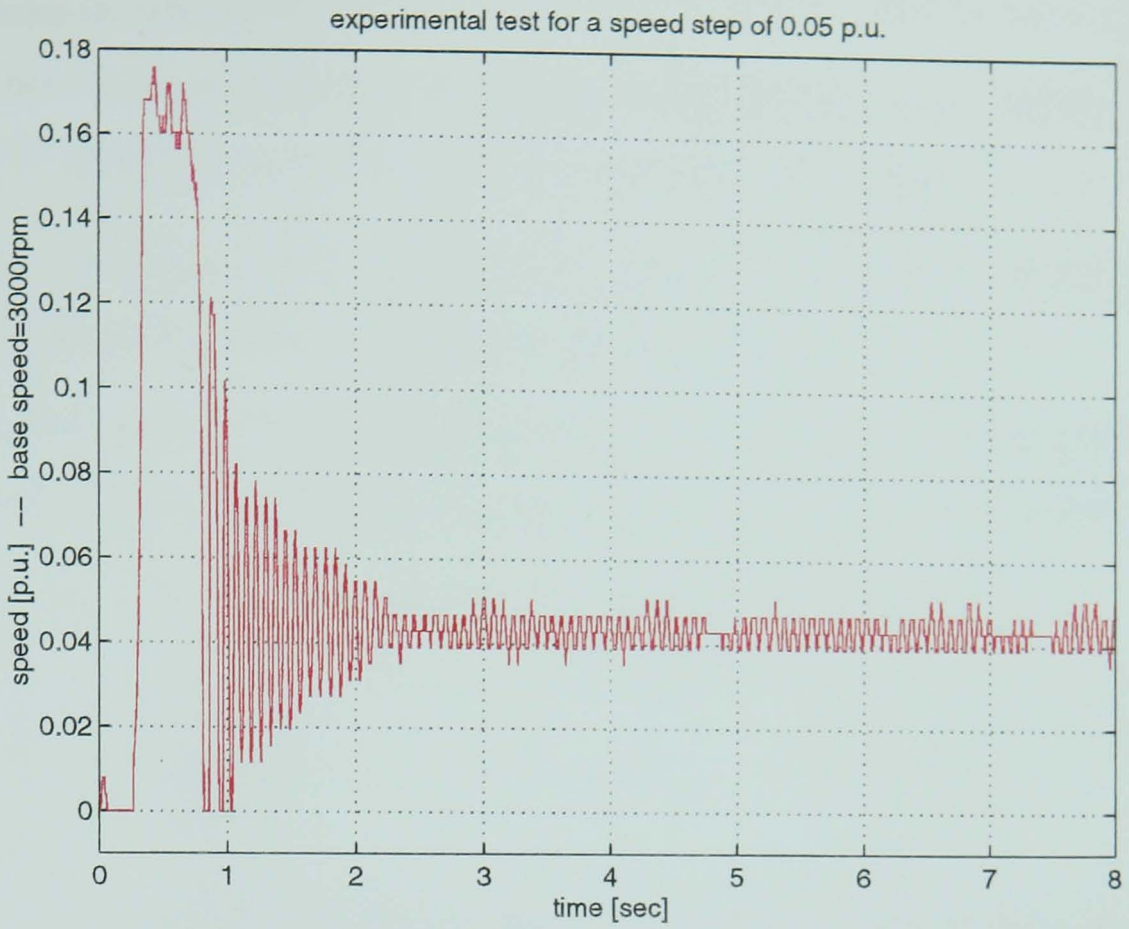

Figure 7.4.1 - The speed response

Figure 7.4.2 presents the v_a voltage plotted against time.

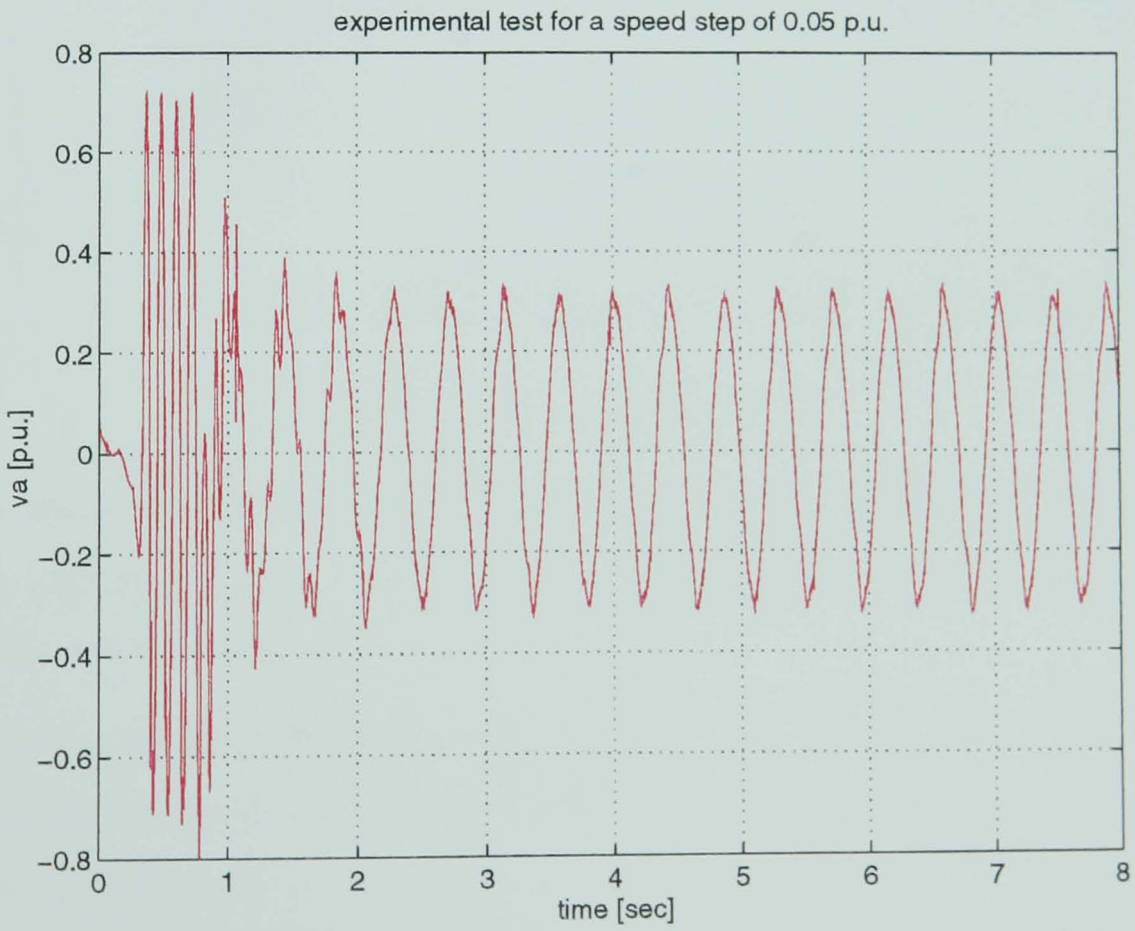

Figure 7.4.2 - the v_a voltage

To effect PWM modulation, the voltages, v_{dref} and v_{qref} are used to calculate the active state times, t_i and t_j [Ref: *(7.2.6)*].The voltages v_a, v_b and v_c are only needed within the

control scheme in order to evaluate the sector at every sampling period. The voltages v_a, v_b and v_c are not measured quantities but they constitute the three-phase balanced system of voltages to be applied to the motor in order to attain the required speed.

Settling time for v_a can be taken as *1.35* seconds and from *2 seconds* onwards, the voltage has reached steady state and has a peak value of about *0.31* p.u.

Figure 7.4.3 show v_a, v_b and v_c plotted together. In the figure they are seen to be a balanced set of voltages which settle to an appropriate sinusoidal form within *2-3 seconds*.

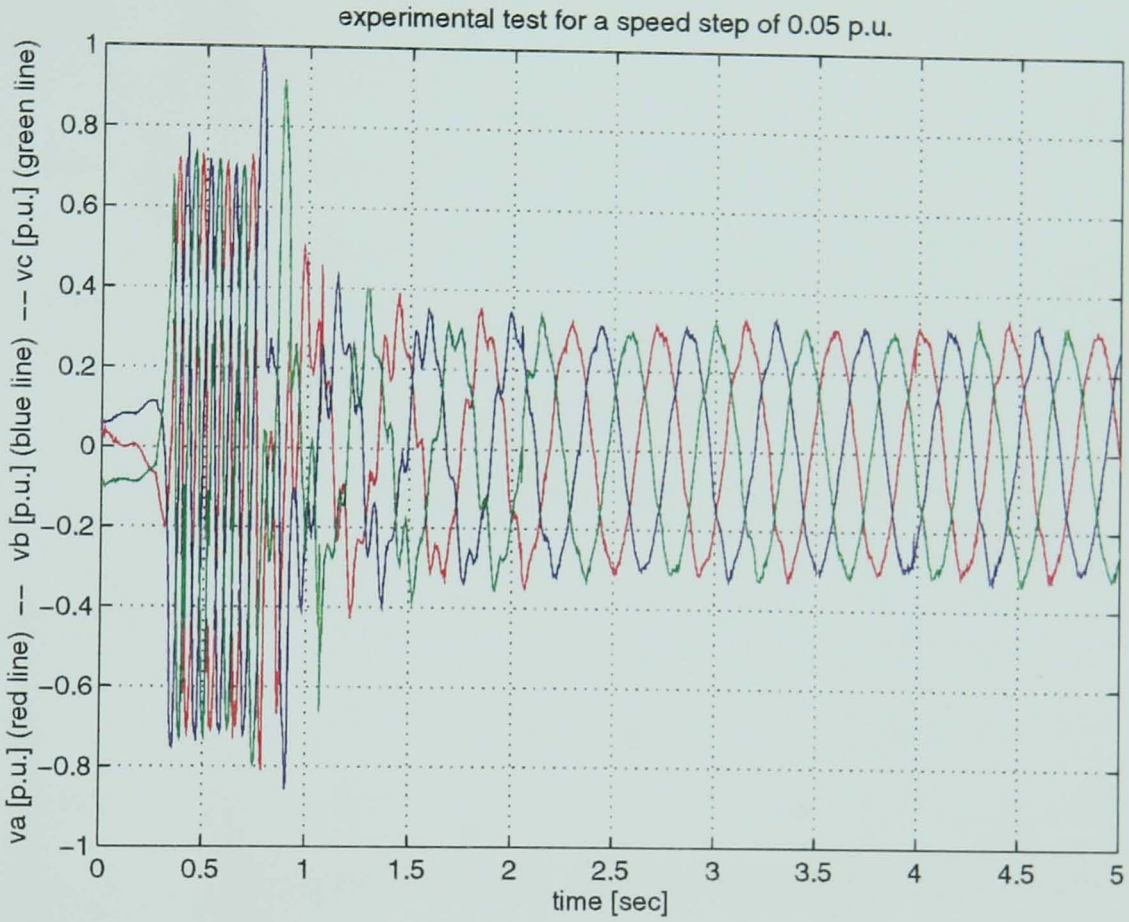

Figure 7.4.3 - v_a, v_b and v_c voltages

Figure 7.4.4 shows the time/im current plot for the same test. Referring to *Figure 7.1.1* it is seen to be representing the output of the `Flux Model`. This current is continuously compared to a reference magnetising current.

The speed step was applied to the motor which is already energised, therefore there is already a level of magnetising current which is maintained at the reference level by a set of d.c. currents applied to the motor. The line currents measured before applying the speed step are approximately around *2 A*, *-1A*, *-1 A* respectively. The *x*-axis component of the stator current has the task of keeping the magnetising current at its reference value. At standstill, and with d.c currents applied to the motor, i_{y_ref} and consequently, v_{y_ref} [Ref: *Figure 7.1.1.*] are 0 and remain at this level until the speed step occurs.

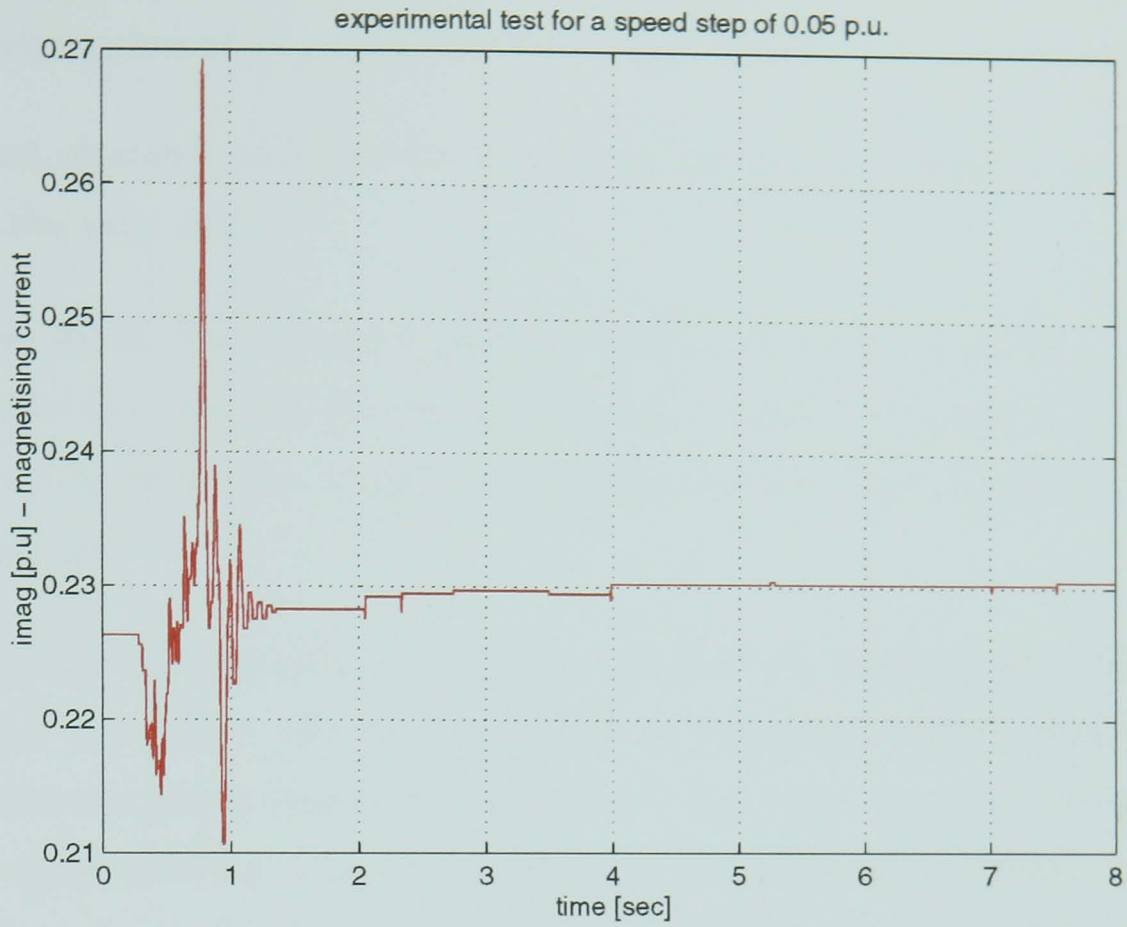

Figure 7.4.4 - magnetising current

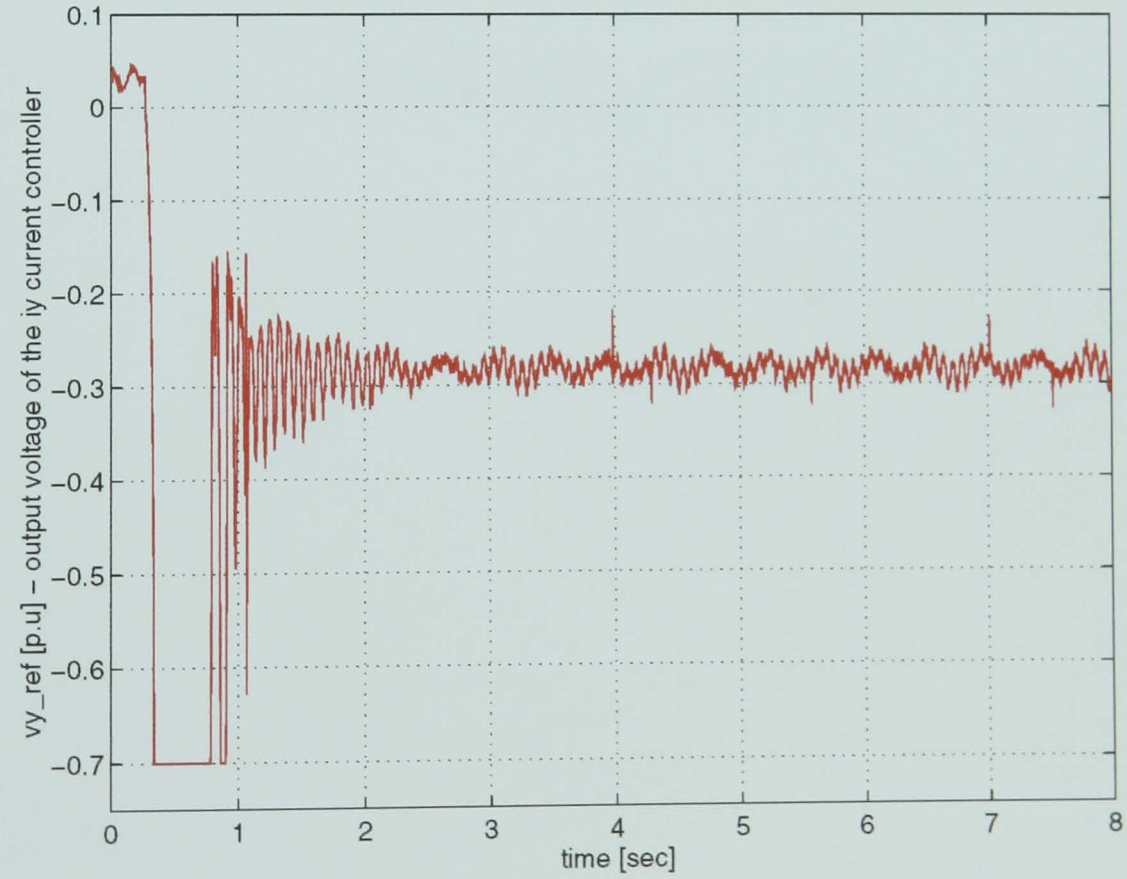

Figure 7.4.5 - output voltage of the i_y current controller

In *Figure 7.4.5* the response of v_{y_ref} is presented. It is seen that as speed increases to around *0.17 p.u.*, the i_y current regulator goes into limitation and v_{y_ref} equals *-0.7 p.u.*, the minimum limit for v_{y_ref} . This causes the speed to decrease and the level of v_{y_ref}

eventually to stabilise at around *-0.3 p.u.* The response of the system is seen to be relative fast and settling time of v_{y_ref} is quantitatively found around *1.3 seconds* as for v_a voltage.

The control algorithm maintains the magnetising current at a constant level, analogously behaving like a d.c. drive.

Apart from speed, other measured quantities that can be used to evaluate control scheme behaviour are the measured line currents. Only two of them are measured by Hall sensors and *Figure 7.4.6* shows one of the two measured currents while *Figure 7.4.7* shows them plotted together.

At time = *0 seconds*, it is seen that the line currents are d.c currents with values: $i_1 = 0.23$ *p.u.* and $i_2 = -0.12$ *p.u.*, and this corresponds to zero speed original condition and then they become alternating currents at about *0.3 p.u.* peak value corresponding to the higher values of speed (*0.16 p.u.*, *Figure 7.4.1*). When the desired speed is reached , currents have also reached the steady state at a peak value of *0.755 p.u.* The high current level is indicative of low speed operation in induction motor drives and is therefore expected.

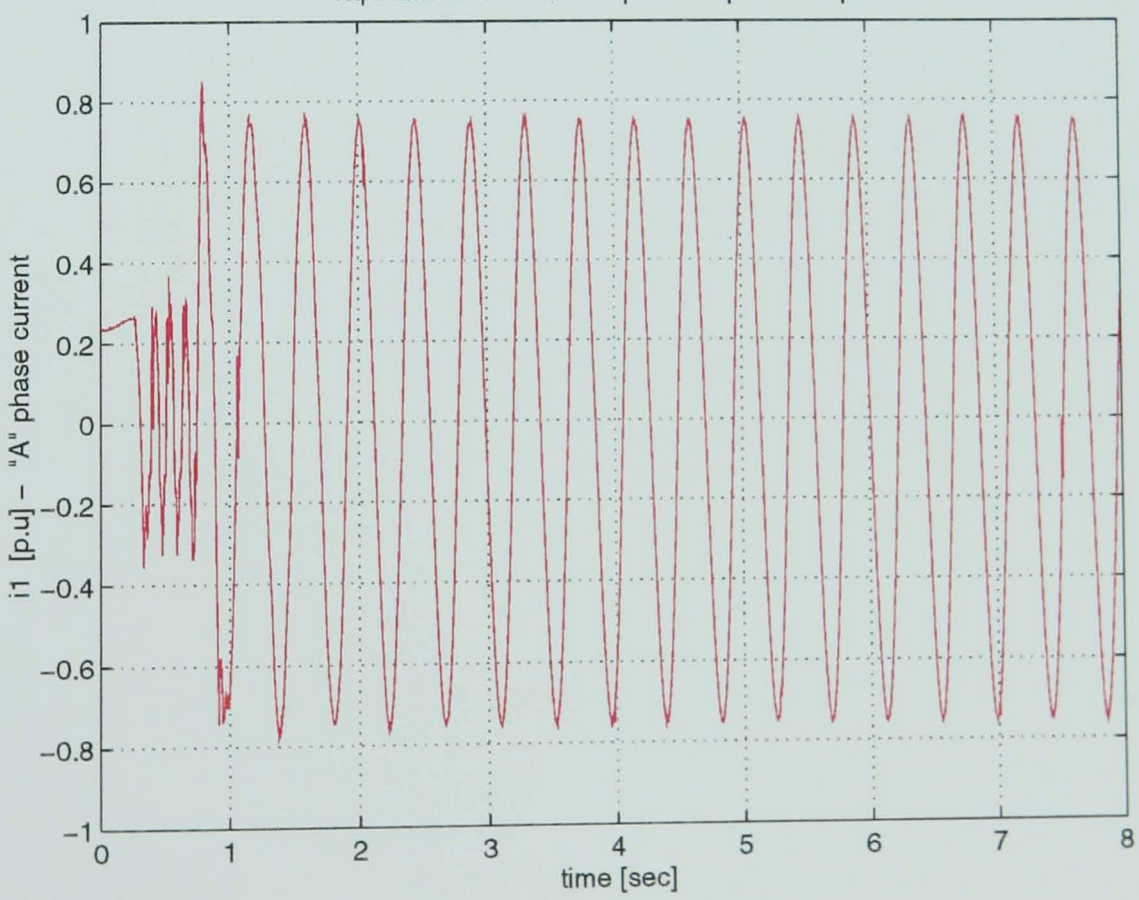

Figure 7.4.6 - measured i1 current

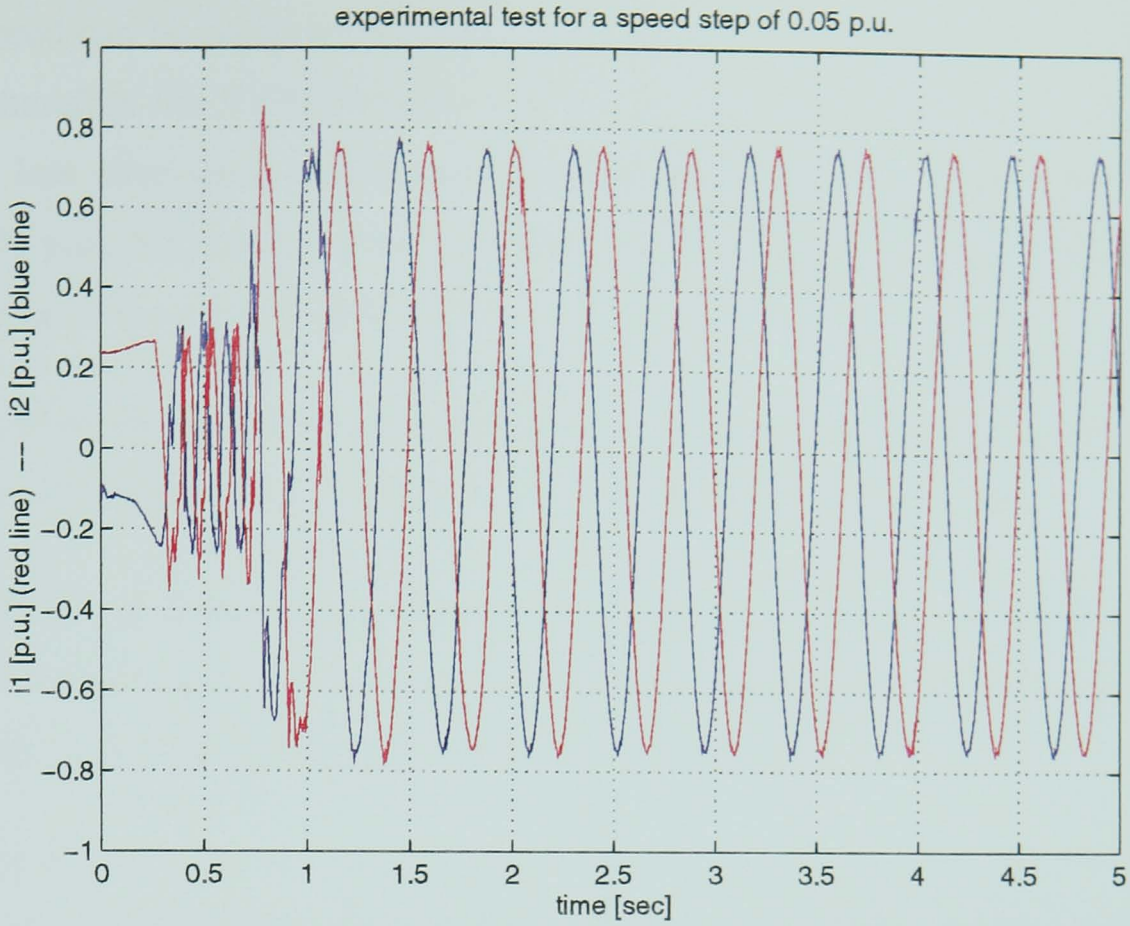

Figure 7.4.7 - measured I1 and I2 currents

In the graph shown in *Figure 7.4.8* another variable, the sector number is taken from DSP memory and plotted.

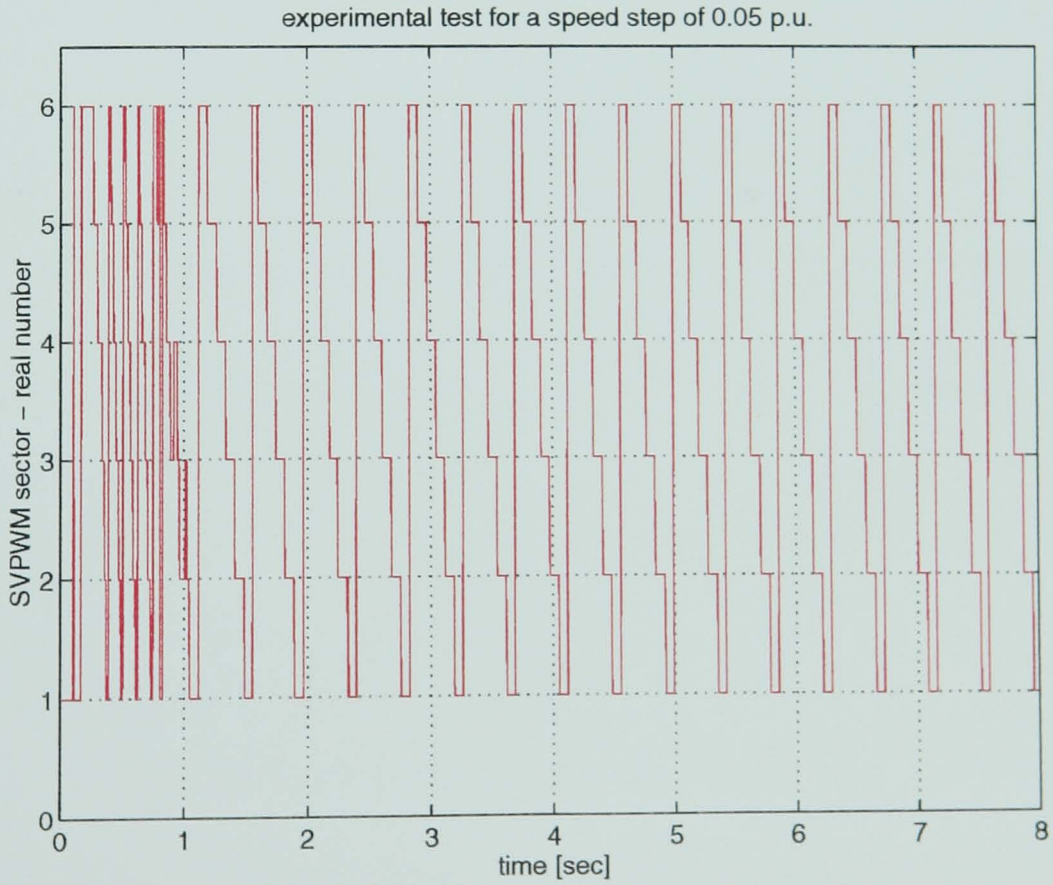

Figure 7.4.8 - SVPWM sectors

With reference to *Figure 7.2.3*, the sector number is plotted against time and has the sequence *6-5-4-3-2-1*. From this information the behaviour of the control scheme at two

levels of speed can be comprehended. In the beginning of the period the speed is around *0.17 p.u.*, therefore the SVPWM frequency is high and a full rotation of *360 degrees* is realised in less time then when speed slows down. When the motor speed stabilises towards *0.05 p.u.*, the rotation is slower as shown. From the plot in *Figure 7.4.8* it can be seen that a clockwise rotation occurs.

The values of machine parameters required in the control unit were maintained constant and precalculated from standard machine no-load and short-circuited tests.

The test system and designed controller are seen to operate correctly and to this extent the objective of building a motor drive test set, demonstrating the validity of its performance and effecting confirmating measurements has been achieved.

The investigation should be continued as all necessary equipment and the assembler implemented control scheme are available. Behaviour of the controller is considered satisfactory, especially as it is stable under low speed conditions which represents a critical operation condition.

Chapter 8

Conclusions and further work

A model has been developed to simulate an induction motor drive. The Simulink toolbox within MATLAB software has been employed for this task. The modelling started with the development of a model for an induction motor, this model being based on DQ-axis theory of the general machine applied for an induction motor.

Subsequently a model for an inverter employing the Pulse-Width Modulation to synthesise a sinusoidal voltage, was achieved.

The two models have been successfully combined to set up a general model of an inverter-fed induction motor. To patch the Simulink diagram of this model, a program was written in MATLAB language. This program offers the possibility of choosing the desired level of harmonics to be considered.

The machine drive model has allowed the analysis of the performance of the drive during the transient period of starting or of sudden changes in load to be effected . The user may choose to feed the induction motor models with harmonics which exhibit an amplitude higher than a certain percentage.

The program allows the change of the motor parameters and simulation data and has the advantage of automatically drawing the drive model, with as many harmonic-fed motor models, as desired. Therefore the user is offered a graphical way to comprehend and analyse the drive, however without having to `drag and drop` blocks in Simulink/MATLAB environment.

By repeating simulations for different levels of load torque or system inertia, results for torque, speed, slip and current are obtained. The results show a good correlation with expected graphs and confirm the very good behaviour of the Simulink developed model.

The modelling of the drive does not take into consideration saturation, therefore the motor inductances are considered constant quantities. This may be considered a deficiency however the effect of saturation may not be important.

Saturation may introduce a third harmonic flux and consequently, third harmonic currents are induced. Still being of low amplitude, they might contribute to the total torque and rotor losses. Saturation can occur either in stator and rotor teeth or in the cores. When teeth saturation is considered, a dominant third harmonic component of the flux is encountered. This third harmonic flux component travels in the air gap with the same velocity and direction as the fundamental air gap flux component, always keeping phase synchronism with the fundamental.

If saturation of the cores is also taken in consideration, then it will have an opposite effect as far as third harmonic flux component is concerned. Saturation of the core sections will produce a third harmonic flux component, which is in direct phase opposition to the third harmonic produced by the saturation of the teeth paths. Therefore by considering both saturation mechanisms, it is considered that the effects given by the third harmonic are cancelled. This reasoning sustains the claim that the induction motor model realised without taking saturation effects into consideration works correctly and that the results will exhibit a low error emanating from this source.

In the past, induction machines have been mostly utilised for fixed-speed application. As billions of new electric motors are responsible for an estimated *70%* consumption of electric power world-wide and DSPs prices have dropped as they become more powerful and sophisticated, industrial applications turn towards induction motors almost exclusively. Control of torque, speed and rotor position may be achieved with high

performance if a vector control method is used. Vector control based on the rotor-flux vector frame is considered in the thesis.

The general indirect field orientation scheme using a speed sensor can also be applied for stator or airgap flux oriented control. However these systems have the disadvantage of needing a shaft angle sensor. If using a shaft sensor is to be avoided then, the flux estimation must be realised from voltage and current measurements. This approach is adopted at low-speeds where high errors given by sensors may be encountered and therefore, a direct field orientation control is favoured. On the other hand, the implementation of a direct flux oriented control system requires more computing power, that is a more powerful and expensive microcontroller. Therefore it was decided to implement an indirect field oriented control scheme with orientation to the rotor-flux vector.

It is well known that indirect rotor-flux control of an induction motor, using a shaft encoder, can provide excellent control when the parameters of the slip gain calculator are known accurately. If speed and currents are correctly measured then the performance is sensitive to the rotor time constant. Therefore the performance of a vector controlled induction motor drive deteriorates when real motor parameters differ from those used by the controller. However the effects of detuning are considered to be more severe in large machines and when a machine operates at reduced flux levels.

As the realisation of performance control is dependent on motor parameter estimation, the work has been carried out to investigate parameter estimation and the sensitivity of the controller to rotor resistance error.

Procedures were developed to estimate total leakage reactance, rotor and stator resistances and rotor time constant of a motor. The algorithms were written in Pascal language to allow simulation. Subsequently, a practical set-up at *the Department of Electrical Engineering, University of L'Aquila , Italy* was employed to perform the motor parameter estimation tests. The results obtained have been compared to motor parameters obtained by classical tests as locked-rotor and no-load tests or given by the manufacturer. Results are reported and overall the procedures have worked satisfactory. Sets of simulation results for three motors and test result for an 1.5 kW induction motor are given. Some errors have been encountered in rotor time constant estimation and a correcting method is proposed.

To analyse the performance of the drive for a variation in rotor resistance for example, simulations were carried out and results shown. For one simulation, the rotor resistance was estimated with a *20%* error. Therefore the erroneous value was introduced in the Pascal simulation program and some tests for a speed step from standstill to 0.1 p.u. and from standstill to 0.4 p.u. and back to 0.1 p.u. were done. The speed response does not exhibit any visible difference proving a low sensitivity of the control scheme to rotor resistance variation. This conclusion which contradicts statements by other researchers, permits the estimated motor parameters to be included in a vector control scheme even when they exhibit errors when compared with the available motor data.

A vector controlled induction motor test-rig was practically realised at *Department of Electrical Engineering, De Montfort University*. This was designed by the author and built within the *Mechanical Department*. The test-rig is based on a *1.5 kW, 50 Hz* squirrel-cage motor. The control strategy is implemented using a DSP-based Evaluation Module from *Texas Instruments Inc.* The inverter used is an experimental one, built within the Department, based on an IGBT module and has a separate driver circuit which raise the PWM signals from 5V to 15 V required to drive the IGBTs.

In order to implement closed loop digital control, the motor phase currents need to be sensed and digitised and Hall-effect sensors are used for this task. The sensing of all three currents is not necessary because the three currents sum to zero due to the unconnected machine neutral, therefore only two sensing circuits were installed.

Noise problems were encountered when measuring the motor phase currents. The output signals from the Hall sensors are sent via ribbon-cable to an interface circuit with the DSP board. The signals have a certain amount of noise when they arrive in the A/D converter and this was identified as a problem. One way to remove some of the noise and increase the resolution is to wind the line cable four times around the Hall sensor mount.

The experimental set-up was tested and results presented in *Chapter 7*. Speed response is shown together with other variables. The phase voltages recorded prove that a three-phase balanced set of voltages are to be applied to the motor after being PWM modulated. Also line measured currents are shown, and these are sinusoidal and balanced. The magnetising current and the voltage output of the i_y current controller and also the sequence of the SVPWM sectors are presented which allow evaluation of system performance to be carried

out. From the test results it can be considered that the rotor frame vector control system implemented performs correctly. The speed response is shown to be reasonably fast and the system maintains stability even at low speeds.

As a result of the above work, the following conclusions can be drawn:

1. The Simulink toolbox in MATLAB is a powerful tool for motor modelling and simulation. It allows building friendly user graphical models and performing various simulations.

2. The motor model built in Simulink is a valuable tool to analyse the performance of the induction motor in various situations including transient phase.

3. The PWM inverter-fed drive model developed using a number of induction motor models combined together and fed with harmonics also proves a good behaviour and allows the user to choose a certain percentage of harmonics to be considered.

4. Motor parameter estimation procedures were developed, simulated and practically tested. Simulation results for three sets of motor data show adequately estimated parameter values. In the case of rotor time constant estimation when unacceptable errors were encountered, a correcting method was proposed and it proved to work accurately.

5. Simulations have been performed for a vector control scheme working with an erroneous rotor resistance when a speed step was applied. Results prove a low sensitivity of the speed response to variation of the motor rotor resistance.

6. An induction motor test rig was practically realised to allow testing of the vector control scheme. Results for a speed step show a satisfactory response and behaviour of the scheme, however the author proposes some future improvements.

   The objectives set in the abstract of the thesis were accomplished.

8.1 Further work proposed

It is suggested that the motor cables, connecting the inverter and the induction motor, and those conveying the switching signals from the DSP board towards the inverter be

screened, with good earth connection. Also it is recommended that a suppression circuit is installed upstream of the inverter to minimise the effects which may be introduced by spikes coming from the motor.

For protection purposes another current sensor may be placed in the d.c. link of the inverter. A different approach is to use a current sensor in the d.c. link from which all necessary current feedback information can be extracted. Based on the inverter switching commands, at suitable instants, samples of d.c. current are taken. The Space Vector modulation technique described in *Chapter 7* uses two adjacent active states in the hexagon to synthesise the required voltage vector. Thus, in each modulation cycle, for certain intervals, the d.c link represents two different motor line currents.

As nowadays more control schemes are sensorless, it is proposed that the control system developed should be improved by incorporating a *Kalman* filter for speed estimation. The existing DSP-based *Evaluation Module* is powerful enough for sensorless implementation.

References

[1] George Gulalo - " Worldwide production of electric motors: changes, growth, and driving forces " in *Details on Digital Control Systems*, volume I, issue 1, July 1998, *Texas Instruments Inc.*

[2] Denis O'Kelly - " Performance and Control of Electrical Machines " , McGraw - Hill Book Company , 1991, pp. 257-275

[3] Stephen J. Chapman - " Electric Machinery Fundamentals " , second edition, McGraw- Hill International Edition, Electrical Engineering Series 1991, pp. 592

[4] " Guide to the speed control of motors: Open-loop control " - © Schneider Ltd. UK *at* http://www.schneider.co.uk/booklets/aug/aug5.htm

[5] F. Blaschke - " The principle of field-orientation as applied to the new Transvektor closed-loop control system for rotating-field machines " in *Siemens Review* , Vol. 34, pp. 217-220, 1972

[6] Liviu Kreindler, Julio C. Moreira, Antonio Testa, Thomas A. Lipo - " Direct Field Orientation Controller using Stator Phase Voltage Third Harmonic " in *IEEE Trans. Ind. Appl.*, Vol. 30, No. 2, pp. 441, 1994

[7] Satoshi Ogasawara, Hirofumi Akagi, Akira Nabae - " The Generalised Theory of Indirect Vector Control for AC Machines " in *IEEE Trans. Ind. Appl.*, Vol. 24, No. 3, pp. 470 ,1988

[8] Rik W. De Doncker , Donald W. Novotny - " The Universal Field Oriented Controller " in *IEEE Trans. Ind. Appl.*, Vol. 30, No. 1, pp. 92-100, 1994

[9] Olorunfemi Ojo, Madhani Vipin, Ishwar Bhat - " Steady-State Performance Evaluation of Saturated Field Oriented Induction Motors " in *IEEE Transactions on Industry Applications* , Vol. 30, No. 6, pp. 1638, 1994

[10] J. Zdenkovic, Z. Kuljic, N. Pasalic - " Speed Sensorless Drive with Induction Machine based on Natural Field Orientation " in *6th European Conference on*

Power Electronics and Applications EPE '95, 19-21 September 1995 Sevilla, Vol. I, pp. 1.752- 1.757

[11] P. J. Coussens, A. P. Van den Bossche, J. A. Melkebeek - " Non-Linear Field Oriented Control in a Rotor Reference Frame " in *6th European Conference on Power Electronics and Applications EPE '95*, 19-21 September 1995 Sevilla, Vol. I, pp. 1.820- 1.825

[12] Patrick L. Jansen, Robert D. Lorenz - " A Physically Insightful Approach to the Design and Accuracy Assessment of the Flux Observers for Field Oriented Induction Machines Drives " in *IEEE Trans. Ind. Appl.*, Vol. 30, No. 1, pp. 101, 1994

[13] Fang-Zheng Peng, Tadashi Fukao - " Robust Speed Identification for Speed-Sensorless Vector Control of Induction Motors " in *IEEE Transactions on Industry Applications* , Vol. 30, No. 5, pp. 1234, 1994

[14] Sakutaro Nonaka, Yasuhiko Neba - " Current regulated PWM-CSI induction motor drive system without a speed sensor " in *IEEE Trans. Ind. Appl.*, Vol. 30, No. 1, pp. 116, 1994

[15] S. C. Chang, S. N. Yeh - " Current sensorless field-oriented control of induction motors " in *IEEE Proc.-Electr. Power Appl.*, Vol. 143, No. 6, pp. 492, 1996

[16] Young-Real Kim, Seung-Ki Sul, Min-Ho Park - " Speed Sensorless Vector Control of Induction Motor using Extended Kalman Filter " in *IEEE Trans. Ind. Appl.*, Vol. 30, No. 5, pp. 1225, 1994

[17] M. Dalla Mora, C. Manes, F. Parasiliti - " A field-oriented induction motor drive with a non-linear state observer " in *IEEE International Conference on Systems , Man and Cybernetics,* Vol. 5, pp. 71, 1993

[18] F. Profumo, G. Griva, M. Pastorelli, J. Moreira, R. De Doncker - " Universal Field Oriented Controller based on Air Gap Flux sensing via Third Harmonic Stator Voltage " in *IEEE Transactions on Industry Applications*, Vol. 30, No. 2, pp. 448, 1994

[19] C. Manes, F. Parasiliti, M. Tursini - " DSP based field-oriented control of induction motor with a non-linear state observer " in *27th Annual IEEE Power Electronics Specialists Conference,* Vol. 2, pp. 1254, 1996

[20] C. Manes, F. Parasiliti, M. Tursini - " A comparative study of rotor flux estimation in induction motors with a non-linear observer and the extended Kalman filter " in *20th International Conference on Industrial Electronics Control and Instrumentation - Bologna,* Vol. 3, pp. 2149, 1994

[21] S. Wade, M. W. Dunnigan, B. W. Williams - " Improving the accuracy of the rotor resistance estimate for vector-controlled induction machines " in *IEEE Proc.-Electr. Power Appl.*, Vol. 144, No. 5, pp. 285-294, 1997

[22] S. Wade, M. W. Dunnigan, B. W. Williams - " Improvements for Induction Machine Vector Control " in *6th European Conference on Power Electronics and Applications EPE '95*, 19-21 September 1995 Sevilla, Vol. I, pp. 1.542- 1.546

[23] G. Heinemann, W. Leonhard - " Self-tuning Field Orientated Control of an Induction Motor Drive " in *International Power Electronics Conference*, Tokyo 1990, pp 415

[24] Chu Wang, Donald W. Novotny, Thomas A. Lipo - " An Automated Rotor Time Constant Measurement System for Indirect Field-oriented Drives " in *IEEE Transactions on Industry Applications*, Vol. 24 , No. 1, pp. 151-158, 1988

[25] Joachim Holtz, Thomas Thimm - " Identification of the Machine Parameters in a Vector-Controlled Induction Motor Drive " in *IEEE Trans. Ind. Appl.*, Vol. 27, No. 6, pp. 1111, 1991

[26] D. J. Atkinson, P. P. Acarnley, J. W. Finch - " Parameter Identification Techniques for Induction Motor Drives " in *EPE Aachen* 1989, pp. 307

[27] E. B. S. Filho, A. M. N. Lima, C. B. Jacobina - " Parameter Estimation for Induction Machines via Non-Linear Least Squares Method " in *IECON 1991*, Japan, pp. 639

[28] K. Tungpimolrut, Fang-Zheng Peng, T. Fukao - " Robust Vector Control of Induction Motor without Using Stator and Rotor Circuit Time Constants " in *IEEE Transactions on Industry Applications,* Vol. 30, No. 5, pp. 1241, 1994

[29] A. A. Ganji, P. Lataire - " Rotor Time Constant Compensation of an Induction Motor in indirect vector controlled drives " in *6th European Conference on Power Electronics and Applications EPE '95*, 19-21 September 1995 Sevilla, Vol. I, pp. 1.431-1.436

[30] L. A. S. Ribeiro, C. B. Jacobina, A. M. N. Lima - " Parameters and Speed Estimation for Induction Machines based on Dynamic Models " in *6th European Conference on Power Electronics and Applications EPE '95*, 19-21 September 1995 Sevilla, Vol. I, pp. 1.496-1.501

[31] H. -J. Shieh, K. -K. Shyu, F. -J. Liu - " Adaptive estimation of rotor time constant for indirect field-oriented induction motor drive " in *IEE Proc.-Electr. Power Appl.*, Vol. 145, No. 2, pp. 111-118, March 1998

[32] Ying-Yu Tzou, Shore-Ting Yeh, Hua Wu - " DSP-base Rotor Time Constant Identification and Slip gain Auto-tuning for Indirect Vector-Controlled Induction Drives " in *IECON Proceedings* Taiwan 1996, Vol. II , pp 1228

[33] Timothy M. Rowan, Russel J. Kerkman, David Leggate - " A Simple On-line Adaptation for Indirect Field Orientation of an Induction Machine " in *IEEE Transactions on Industry Applications*, Vol. 27, No. 4, July/August 1991, pp. 720

[34] D. Holliday, T. C. Green, B. W. Williams - " On-line parameter measurement of induction machines " in *IECON 1993 Proceedings* ,USA, Vol. II, page 992

[35] H. Kubota, K. Matsue - " Speed Sensorless Field-oriented Control of Induction Motor with Rotor Resistance Adaptation " in *IEEE Transactions on Industry Applications* , Vol. 30, No. 5, pp. 1219, 1994

[36] N. Khenfer, A. Rezzoug, E. J. Gudefin, F. Meibody-Tabar - " Identification of Parameters of Asynchronous Machines. Experimental Methods and Results " in *UPEC Manchester 1992*, pp. 283

[37] H. Schierling - " Self-commissioning - a Novel Feature of Modern Inverter-fed Induction Motor Drives " in *the 3rd Power Electronics and Variable Speed Drives* London 1988, pp 287

[38] M Ruff, H. Grotstollen - " Off-line Identification of the Electrical Parameters of an Industrial Servo Drive System " , 1996 , California ,pp 213

[39] Dong Seong Oh, Kwan Yuhl Cho and Myung Joong Youn - " New rotor Parameter Estimation for a Flux and Speed Control of the Induction Machine considering Saturation Effects " in *IECON 1991* , Japan, Vol. I , pp. 561

[40] M. S. Herbert, A. J. Curley, R. Perryman - " Computer Based Simulation of a Vector Controlled AC Drive System for Performance Analysis and Condition Monitoring "

[41] B. Lemaire-Semail, F. Bouillault, A. Razek - " Modelling of vector controlled cage induction motor with FEM " in *IEEE Proceedings-B* Vol. 138, No. 6, pp. 297, 1991

[42] B. C. Ghosh, S. N. Bhadra - " Effects of flux level on a CSI-fed field-oriented induction motor " in *IEEE Proc.-Electr. Power Appl.*, Vol. 144, No. 5, pp. 295-300, 1997

[43] Katsunori Taniguchi, Yasumasa Ogino, Hisaichi Irie - " PWM Technique for Power MOSFET Inverter " in *IEEE Transactions on Power Electronics*, Vol. 3, No. 3, pp. 328-334, 1988

[44] Hisao Kubota, Kouki Matsue, Jong-Ha Ree - " Analysis of New Current Source GTO Inverter-fed Induction Motor Drive " in *IEEE Transactions on Power Electronics*, Vol. PE-1, No. 4, pp 210-214, 1986

[45] Russel J. Kerkman, Timothy M. Rowan - " Voltage-Controlled Current-Regulated PWM Inverters " in *IEEE Transactions on Industry Applications* , Vol. 26, No. 2, pp. 244-251, 1990

[46] John K. Pedersen, Paul Thϕgersen - " Fully Digital-controlled PWM Inverter with Software based Modulation for AC Machine Control " in *IEEE*, pp. 996 , 1990

[47] A. Haras, D. Roye - " Vector PWM Modulator with Continuous Transition to the Six-Step Mode " in *6th European Conference on Power Electronics and Applications EPE '95*, 19-21 September 1995 Sevilla, Vol. I, pp. 1.729-1.734

[48] V. G. Agelidis, H. C. Goh - " Low-distortion variable-level PWM technique " in *IEE Proc.-Electr. Power Appl.*, Vol. 145, No. 2, pp. 73-78, March 1998

[49] D. Horstmann, G. Stanke - " Field orientated control with subordinated pulse pattern generation of PWM converters fed induction machines " in *IFA6 Software for Computer Control, S. Africa*, 1988

[50] A. M. Walczyna, K. Hasse, R. Czarnecki - " Input Filter Stability of Drives Fed from Voltage Inverters Controlled by Direct Flux and Torque Control Methods " in *IEE Proc.- Electr. Power Appl.,* Vol. 143, No. 5, pp. 396-402, September 1996

[51] I. Takahashi, T. Noguchi - " A new quick response and high-efficiency control strategy of an induction motor " in *IEEE Trans., IA-22 (5),* pp. 820- 827 , 1986

[52] I. Takahashi , Y. Ohmori - " High performance direct torque control of an induction motor " in *IEEE Trans. , IA-25 (2),* pp. 257-264 , 1989

[53] Dong-Choon Lee, Seung-Ki Sul, Min-Ho Park - " High Performance Current Regulator for a Field-Oriented Controlled Induction Motor Drive " in *IEEE Transactions on Industry Applications* , Vol. 30, No. 5, pp. 1247-1257 , 1994

[54] D. M. Brod , D. W. Novotny - " Current control of VSI-PWM inverters " in *IEEE Trans. Ind. Applications*, Vol. IA-21, pp. 562- 570, 1985

[55] T. M. Rowan, R. J. Kerkman - " A new synchronous current regulator and an analysis of current-regulated PWM inverters " in *IEEE Trans. Ind. Applications*, Vol. IA-22, pp. 678- 690, 1986

[56] R. D. Lorenz, D. B. Lawson - " Performance of feedforward current regulators for field-oriented induction machine controllers " in *IEEE Trans. Ind. Applications*, Vol. IA-23, pp. 597- 602, 1987

[57] L. Ben-Brahim, A. Kawamura - " Digital current regulation of field-oriented controlled induction motor based on predictive flux observer " in *IEEE Trans. Ind. Applications*, Vol. 27, pp. 956- 961, 1991

[58] J. -W. Choi, H. -W. Kim, S. -K. Sul - " New current control concept-minimum time current control in induction machine drive " in *Conf. Rec. IEEE IECON*, pp. 311- 316, 1995

[59] Dae-Woong Chung, Seung-Ki Sul - " Analysis and Compensation of Current Measurements Error in Vector-Controlled AC Motor Drives " in *IEEE Transactions on Industry Applications,* Vol. 34, No. 2, pp 340-345, 1998

[60] B. J. Seibel , T. M. Rowan, R. J. Kerkman - " Field-Oriented Control of an Induction Machine in the Field-Weakening Region with DC-Link and Load Disturbance Rejection " in *IEEE Transactions on Industry Applications,* Vol. 33, No. 6, pp 1578-1584 , 1997

[61] L. Umanand, S. R. Bhat - " Optimal and robust digital current controller synthesis for vector-controlled induction motor drive systems " in *IEEE Proc.-Electr. Power Appl.*, Vol. 143, No. 2, pp. 141-150, 1996

[62] C. Attaianese, A. Damiano, I. Marongiu, A. Perfetto - " Total Digital Vectorial Control of an Asynchronous Motor: a Prototypical Realisation "

[63] C. Attaianese, A. Damiano, I. Marongiu, A. Perfetto - " Robust model reference control of induction motor drive " in *Conf. Proceedings of Power Electronics Motion Control*, Warsaw (Poland), 1994

[64] E. K. K. Sng, A. C. Liew - " On Line Tuning of Rotor Flux Observers for Field Oriented Drives Using Improved Stator Based Flux Estimator For Low Speeds " in *6th European Conference on Power Electronics and Applications EPE '95*, 19-21 September 1995 Sevilla, Vol. I, pp. 1.437- 1.442

[65] J. K. Al-Tayie, P. P. Acarnley - " Estimation of speed, stator temperature and rotor temperature in cage induction motor drive using the extended Kalman filter algorithm " in *IEEE Proc.-Electr. Power Appl.*, Vol. 144, No. 5, pp. 301-309, 1997

[66] Raymond B. Sepe, Jeffrey H. Lang - " Implementation of Discrete-Time Field-Oriented Current Control " in *IEEE Trans. Ind. Appl.*, Vol. 30, No. 3, pp. 723, 1994

[67] T. L. Chern, C. S. Liu, C. F. Jong, G. M. Yan - " Discrete integral variable structure model following control for induction motor drivers " in *IEEE Proc.-Electr. Power Appl.*, Vol. 143, No. 6, pp. 467, 1996

[68] Peter Vas - " Vector Control of AC Machines " , Clarendon Press - Oxford , 1990, pp. 31, 1-4, 23-27, 305-317

[69] S. B. Dewan, G.R. Slemon, A. Straughen - " Power Semiconductor Drives ", a Wiley-Interscience Publication, John Wiley & Sons, 1984, pp. 3-5

[70] " Matlab - The Language of Technical Computing " by The MATHWORKS Inc. *at* http://www.mathworks.com/products/matlab/

[71] " Simulink 2.2 " by The MATHWORKS Inc. *at* http://www.mathworks.com/products/simulink/#overview

[72] Amar Bousbaine - " An investigation into the thermal modelling of induction motors ", a PhD thesis submitted to the University of Sheffield, Department of Electronic and Electrical Engineering, June 1993

[73] Dennis O'Kelly and S. Simmons - " Introduction to Generalized Electrical Machine Theory " , McGraw-Hill ,1968

[74] S. R. Bowes, R. R. Clements - " Computer-aided design of PWM inverter systems " in *IEE Proceedings~B Electric Power Applications*, Vol. 129, No. 1, January 1982, pp. 1-17

[75] S. R. Bowes, J. C. Clare - " Computer-aided design of PWM power-electronic variable-speed drives " in *IEE Proceedings~B Electric Power Applications,* Vol. 135, No. 5, September 1988, pp. 240-260

[76] A. I. Maswood - " Computer application in the analysis of rectifier and inverters " in *IEE Proceedings~B Electric Power Applications*, Vol. 142, No. 4, July 1995, pp. 233-238

[77] G. Franceschini, S. Pirani, M. Rinaldi, C. Tassoni - " Spice-Assisted Simulation of Controlled Electric Drives: An Application to Switched Reluctance Drive " in *IEEE Transactions on Ind. Applications,* Vol. 27, No. 6, November/December 1991, pp. 1103-1110.

[78] V. Rajagopalan - " Computer-aided analysis of power electronic systems " - published by Marcel Dekker Inc., 1987

[79] J. A. Martinez-Velasco, N. Mohan - " ATP Simulation of Power Electronics Systems using a Data-Modula approach " in *Proceedings of the 32nd UPEC '97*, Vol. 1 , pp. 495-498

[80] B. Baha - " Modelling of resonant switched-mode converters using SIMULINK " in *IEE Proceedings~B Electric Power Applications*, Vol. 145, No. 3, May 1998, pp. 159-163

[81] Cyril W. Lander - " Power Electronics " , McGraw - Hill Book Company, 1993, pp. 204, 400-401, 295-297.

[82] P. G. Handley, Prof. J. T. Boys - " Practical real-time PWM modulators: an assessment " in *IEE Proceedings~B Electric Power Applications*, Vol. 139, No.2, March 1992, pp. 96-102

[83] Robert D. Ramirez - " The FFT Fundamentals and Concepts ", Prentice Hall, 1985

[84] M. McCormick - " Losses in Variable Speed Induction Motors Supplied with PWM Waveforms " in *UPEC Proceedings*, University of Dundee

[85] " Space Vector Modulation " - © 1995-1998 Analog Devices, Inc. *at* http://www.analog.com/industry/motor_control/seminars/mctech/ch2.html

[86] " Bus Clamping forms of SVM " - © 1995-1998 Analog Devices, Inc. *at* http://www.analog.com/industry/motor_control/seminars/mctech/ch4.html

General Bibliography

Papers

[87] C. Manes, F. Parasiliti, M. Tursini - " Osservatore non lineare per la stima dello stato di un sistema azionamento-carico " in *Seminario Interattivo su Azionamenti Elettrici con Controllo a Microprocessore*, Bressanone, 1994

[88] F. Parasiliti, M. Tursini, D. Zhang - " On-line Self-Tuning of PI Controllers for High Performance PMSM Drives " in *Conference Record of 31st IAS Annual Meeting*, Vol. 3, pp. 1619-1625, 1996

[89] J. G. Cho - " IGBT based Zero Voltage Transition Full Bridge PWM Converter for High Power Applications " in *IEE Proc. -Electr. Appl.*, Vol. 143, No. 6 , pp 475-480, 1996

[90] C. -M. Young, C. -C. Liu, C. -H. Liu - " New inverter-driven design and control method for two-phase induction motor drives " in *IEE Proc.-Electr. Power Appl.*, Vol. 143, No. 6, pp. 458-466 , 1996

[91] Milutin G. Jovanovic, Robert E. Betz, Don Platt - " Sensorless Vector Controller for a Synchronous Reluctance Motor " in *IEEE Transactions on Industry Applications,* Vol. 34, No. 2, pp 346-354, 1998

[92] J. Hunter - " Motor Control: Why dsps ? " in *New Electronics on campus spring 1998*, pp. 30-31

[93] S. Ertem , Y. Baghzouz - " A Fast Recursive Solution for Induction Motor Transients " in *IEEE Trans. Ind. Appl.*, Vol. 24, No. 5, pp. 558, 1988

[94] Robert D. Lorenz, Donald W. Novotny - " Saturation Effects in Field-Oriented Induction Machines " in *IEEE Trans. Ind. Appl.*, Vol. 26, No. 2, pp. 283 ,1990

[95] Thomas G. Habetler, Francesco Profumo, Michele Pastorelli, Leon M. Tolbert - " Direct Torque Control of Induction Machines using Space Vector Modulation " *IEEE 1991-0-7803-0453-5/91*

[96] Joachim Holtz, Eckhard Bube - " Field-Oriented Asynchronous Pulse-Width Modulation for High-Performance ac Machine Drives operating at Low Switching Frequency " in *IEEE Trans. Ind. Appl.*, Vol. 27, No. 3, pp. 574, 1991

[97] Thomas R. Doll, Andrew H. Kaiser - " Vector Controls " in *IEEE*, pp. 59 , 1992

Books

[98] R. Langlois-Berthelot - " Electro-magnetic Machines ", MacDonald & Co. , 1953

[99] Syed A. Nasar - " Electric Machines and Power Systems " , Volume 1 , "Electric Machines " , McGraw - Hill , Inc. , 1995

[100] W. Bolton - " Fourier Series " , Longman Scientific & Technical

[101] Richard M. Crowder - " Electric Drives and their Controls " , Clarendon Press , Oxford , 1995

[102] T. Mohan, T.M. Undeland, W.P. Robbins - " Power Electronics: Converters, Applications and Design ", John Wiley & Sons , 1989

[103] W. Leonhard - " Control of Electrical Drives " , Springer-Verlag , 1996

[104] MATLAB - High-Performance Numeric Computation and Visualization Software, *User's Guide* , February 1993, by The MathWorks, Inc.

[105] MATLAB - High-Performance Numeric Computation and Visualization Software, *External Interface Guide* , February 1993, by The MathWorks, Inc.

[106] SIMULINK - Dynamic System Simulation Software, *User's Guide* , April 1993, by The MathWorks, Inc.

[107] TMS320C24X DSP Controllers - Reference Set, Volume 1: CPU, System and Instruction Set, © 1997, Texas Instruments Incorporated.

[108] TMS320C24X DSP Controllers - Reference Set, Volume 2: Peripheral Library and Specific Devices, © 1997, Texas Instruments Incorporated.

[109] TMS320C2XX - C Source Debugger - User's Guide, © 1997, Texas Instruments Incorporated.

[110] TMS320C1X/C2X/C2XX/C5X - Assembly Language Tools - User's Guide, © 1997, Texas Instruments Incorporated.

Publications:

[1] A. Novinschi, M. McCormick, W. F. Low - " Accurate Simulation of the Induction Motor Drive " *in 32nd Universities Power Engineering Conference Proceedings, UPEC '97*, 10-12 September 1997 Manchester, Vol. 1, pp. 198-201

A paper accepted for *the Fifth International Conference on Rotating Machines*, January 1999, Mumbai, India :

[2] A. Novinschi, M. McCormick, W. F. Low - " A Novel Simulation Methodology For Induction Motor Drives "

APPENDIX 1 - *Motor parameter data file*

The MATLAB *.m* file (*/home/anca/mmatlab/mot/dateind.m*) containing data from motor parameters for running various simulations. For example, the Simulink diagram from *Figure 3.2.2* is simulated with this set of motor parameters.

```
RD=5;
RQ=5;
Rd=3.52;
Rq=3.52;
LD=0.5687;
LQ=0.5687;
Ld=0.5687;
Lq=0.5687;
MdD=0.5411;
MDd=0.5411;
MqQ=0.5411;
MQq=0.5411;
M=0.5411;
GqD=0.5411;
GdQ=-0.5411;
Gqd=0.5687;
Gdq=-0.5687;
pp=2
J=0.005
D=0
Tl=0.1
```

APPENDIX 2 - *Matlab functions*

This appendix contains functions written by the author in the MATLAB language. These functions allow the Simulink blocks developed by the author and presented in *Chapter 3* and *4* to operate.

1. An function:

```
function [An]=An(fq,v,vsin,tri,nr)
ac=acoef(fq,v,vsin,tri,nr);
f=faza(fq,v,vsin,tri,nr);
bc=bcoef(fq,v,vsin,tri,nr);
for cont=1:nr
        a=ac(cont)/sin(f(cont));
        An=[An a];
end;
An=reshape(An,length(An),1);
```

2. acoef function:

```
function [acoef]=acoef(fq,v,vsin,tri,nr)
r=rez(fq,v,vsin,tri);
n=1;
ae=0;
while n<=(nr*2-1)
        for inc=2:2:(2*tri-1)
                ti=2*pi*fq*r(inc);
                tii=2*pi*fq*r(inc+1);
                a=2*v*(sin(n*tii)-sin(n*ti))/(pi*n);
                ae=ae+a;
        end
        acoef=[acoef ae];
        ae=0;
        n=n+2;
end;
acoef=reshape(acoef,length(acoef),1);
```

3. bcoef function:

```
function [bcoef]=bcoef(fq,v,vsin,tri,nr)
r=rez(fq,v,vsin,tri);
n=1;
be=0;
while n<=(nr*2-1)
        for inc=2:2:(2*tri-1)

                ti=2*pi*fq*r(inc);
                tii=2*pi*fq*r(inc+1);
                b=2*v*(cos(n*ti)-cos(n*tii))/(pi*n);
                be=be+b;
        end
        bcoef=[bcoef be];
        be=0;
        n=n+2;
end;
bcoef=reshape(bcoef,length(bcoef),1);
```

4. faza function:

```
function [faza]=faza(fq,v,vsin,tri,nr)
ac=acoef(fq,v,vsin,tri,nr);
bc=bcoef(fq,v,vsin,tri,nr);
for cont=1:nr
        f=atan((ac(cont))/bc(cont));
        faza=[faza f];
end;
faza=reshape(faza,length(faza),1);
```

5. vpoz function:

```
function [vpoz]=vpoz(fq,v,vsin,tri)
m=mo(fq,v,vsin,tri);
indic=(length(m)-4)/4;
vpoz=[0 0 ];
for i=1:indic
    vpoz=[vpoz 0 v v 0];
end
vpoz=[vpoz 0 0 ];
vpoz=reshape(vpoz, length(vpoz), 1);
```

6. vneg function:

```
function [vneg]=vneg(fq,v,vsin,tri)
m=mo(fq,v,vsin,tri);
indic=(length(m)-4)/4;
vneg=[0 0];
for i=1:indic
    vneg=[ vneg 0 -v -v 0];
end
vneg=[vneg 0 0 ];
vneg=reshape(vneg,length(vneg), 1);
```

7. valori function:

```
function [valori]=valori
 valori=[ 0 2 2 0 -2 -2 0];
 valori=reshape(valori, length(valori),1);
```

8. timp function:

```
function [timp]=timp(fq)
 timp=[ 0 0 1/(fq*2)  1/(fq*2)  1/(fq*2)  1/fq 1/fq];
 timp=reshape(timp, length(timp),1);
```

9. mm program:

```
function [ret,x0,str,ts,xts]=mm(t,x,u,flag);
%MM     is the M-file description of the SIMULINK system named MM.
%       The block-diagram can be displayed by typing: MM.
%
%       SYS=MM(T,X,U,FLAG) returns depending on FLAG certain
%       system values given time point, T, current state vector, X,
%       and input vector, U.
%       FLAG is used to indicate the type of output to be returned in SYS.
%
%       Setting FLAG=1 causes MM to return state derivatives, FLAG=2
%       discrete states, FLAG=3 system outputs and FLAG=4 next sample
%       time. For more information and other options see SFUNC.
%
%       Calling MM with a FLAG of zero:
%       [SIZES]=MM([],[],[],0),  returns a vector, SIZES, which
%       contains the sizes of the state vector and other parameters.
%               SIZES(1) number of states
%               SIZES(2) number of discrete states
%               SIZES(3) number of outputs
%               SIZES(4) number of inputs
```

```
%                SIZES(5) number of roots (currently unsupported)
%                SIZES(6) direct feedthrough flag
%                SIZES(7) number of sample times
%
%        For the definition of other parameters in SIZES, see SFUNC.
%        See also, TRIM, LINMOD, LINSIM, EULER, RK23, RK45, ADAMS, GEAR.

% Note: This M-file is only used for saving graphical information;
%       after the model is loaded into memory an internal model
%       representation is used.

% the system will take on the name of this mfile:

nr=input('please type how many harmonics you want to search :') ;
proc=input('please type the value of procentage you want to be considered:');
answer=input('do you want to type the values y/n ?','s');
a='y';
rez=strcmp(a,answer);

if rez==1
        fq=input('please type the value of frequency: ');
        v=input('please type the value of voltage : ');
        vsin=input('please type the max value for sin  :');
        tri=input('please type the number of triungular waves in half a cycle :  ');
else
        fq=50
        v=508.27
        vsin=508.27
        tri=6
end

disp('Please wait..')

amplit=An(fq,v,vsin,tri,nr);
phase=faza(fq,v,vsin,tri,nr);
f=freq(fq,nr);

clear newamp;
clear newphase;
clear newfq;

n=1;
for cont=2:nr
        nn=2*cont-1;
        n=[n nn];
end
n=reshape(n,length(n),1);

cnt=0;
for cont=1:nr
        if abs(amplit(cont))>=proc/100*amplit(1)
                cnt=cnt+1;
                newamp(cnt)=amplit(cont);
                newphase(cnt)=phase(cont);
                newfq(cnt)=f(cont);
                newdeg(cnt)=n(cont);
        end
end

newamp =reshape(newamp,length(newamp),1);
newphase=reshape(newphase,length(newphase),1);
newfq=reshape(newfq,length(newfq),1);
newdeg=reshape(newdeg,length(newdeg),1);

nr=cnt;

dateind
```

```
disp('we have just loaded the data..')
answer=input('would you like to change some data ?  y/n.. ','s');
a='y';
rz=strcmp(a,answer);
if rz==1
        disp('the torque load is:')
        disp(Tl)
        disp('..you can change it..')
        Tl=input('please type the new value for the torque load:')
end

disp('now it is time to draw the SIMULINK system..which will have:')
disp(nr)
disp('induction motor models..')

sys = mfilename;
new_system(sys)
simver(1.3)
if (0 == (nargin + nargout))
   set_param(sys,'Location',[20,2,650,70*nr])
   open_system(sys)
end;
set_param(sys,'algorithm',     'RK-45')
set_param(sys,'Start time',    '0.0')
set_param(sys,'Stop time',     '1')
set_param(sys,'Min step size', '0.001')
set_param(sys,'Max step size', '0.01')
set_param(sys,'Relative error','1e-3')
set_param(sys,'Return vars',   '[t,x,y]')

add_block('built-in/Outport',[sys,'/','Out_torq'])
set_param([sys,'/','Out_torq'],...
                'Port',nr+1,...
                'position',[350,70.5+32.5*(nr-1),370,90.5+32.5*(nr-1)])

add_block('built-in/Sum',[sys,'/','Sum'])
set_param([sys,'/','Sum'],...
                'inputs',nr,...
                'position',[270,3,290,74+65*(nr-1)])

add_line(sys,[295,36.5+32.5*(nr-1);330,36.5+32.5*(nr-1);330,78.5+32.5*(nr-1);345,78.5+32.5*(nr-1)])

add_block('built-in/Constant',[sys,'/','Tload'])
set_param([sys,'/','Tload'],...
                'Font Size',12,...
                'Value',Tl,...
                'position',[310,(2.5)+32.5*(nr-1),330,(22.5)+32.5*(nr-1)])

%    Subsystem  'ecmec'.

new_system([sys,'/','ecmec'])
set_param([sys,'/','ecmec'],'Location',[60,20644269,663,20644548])

add_block('built-in/Sum',[sys,'/','ecmec/Sum4'])
set_param([sys,'/','ecmec/Sum4'],...
                'Font Size',12,...
                'inputs','--+',...
                'position',[465,120,490,180])

add_block('built-in/Gain',[sys,'/','ecmec/Gain17'])
set_param([sys,'/','ecmec/Gain17'],...
                'Font Size',12,...
                'Gain',D,...
                'position',[355,111,390,149])

add_block('built-in/Note',[sys,'/','ecmec/speed'])
set_param([sys,'/','ecmec/speed'],...
```

```
                    'Font Size',12,...
                    'position',[285,100,290,105])

add_block('built-in/Gain',[sys,'/','ecmec/Gain16'])
set_param([sys,'/','ecmec/Gain16'],...
                    'Font Size',12,...
                    'Gain',1/J,...
                    'position',[95,109,140,151])

add_block('built-in/Integrator',[sys,'/','ecmec/Integrator4'])
set_param([sys,'/','ecmec/Integrator4'],...
                    'Font Size',12,...
                    'position',[200,112,225,148])

add_block('built-in/Outport',[sys,'/','ecmec/out_1'])
set_param([sys,'/','ecmec/out_1'],...
                    'position',[290,250,310,270])

add_block('built-in/Inport',[sys,'/','ecmec/in_1'])
set_param([sys,'/','ecmec/in_1'],...
                    'position',[15,45,35,65])

add_block('built-in/Inport',[sys,'/','ecmec/in_2'])
set_param([sys,'/','ecmec/in_2'],...
                    'Port','2',...
                    'position',[405,235,425,255])
add_line([sys,'/','ecmec'],[430,245;440,245;440,170;460,170])
add_line([sys,'/','ecmec'],[145,130;195,130])
add_line([sys,'/','ecmec'],[395,130;460,130])
add_line([sys,'/','ecmec'],[495,150;505,150;505,205;65,205;65,130;90,130])
add_line([sys,'/','ecmec'],[230,130;350,130])
add_line([sys,'/','ecmec'],[230,130;285,130;285,260])
add_line([sys,'/','ecmec'],[40,55;370,70;368,72;423,72;430,75;440,75;440,150;460,150])

%    Finished composite block 'ecmec'.

set_param([sys,'/','ecmec'],...
                    'position',[350,32.5*(nr-1),380,55+32.5*(nr-1)])

add_line(sys,[295,36.5+32.5*(nr-1);345,36.5+32.5*(nr-1)])

add_line(sys,[335,10.5+32.5*(nr-1);345,10.5+32.5*(nr-1)])

add_line(sys,[385,25+32.5*(nr-1);400,25+32.5*(nr-1);400,90+65*(nr-1);85,90+65*(nr-1);85,55])

%    Subsystem 'ecslip'.

new_system([sys,'/','ecslip'])
set_param([sys,'/','ecslip'],'Location',[5,22872601,435,22872752])

add_block('built-in/Constant',[sys,'/',['ecslip/syncspeed',13,'']])
set_param([sys,'/',['ecslip/syncspeed',13,'']],...
                    'Font Size',12,...
                    'Value',2*pi*fq/pp,...
                    'position',[60,52,145,88])

add_block('built-in/Sum',[sys,'/','ecslip/Sum6'])
set_param([sys,'/','ecslip/Sum6'],...
                    'Font Size',12,...
                    'inputs','+-',...
                    'position',[215,59,240,101])

add_block('built-in/Gain',[sys,'/','ecslip/Gain20'])
set_param([sys,'/','ecslip/Gain20'],...
                    'Font Size',12,...
                    'Gain',pp/(2*pi*fq),...
                    'position',[285,57,350,103])
```

```
add_block('built-in/Outport',[sys,'/','ecslip/out_1'])
set_param([sys,'/','ecslip/out_1'],...
                    'position',[380,70,400,90])

add_block('built-in/Inport',[sys,'/','ecslip/in_1'])
set_param([sys,'/','ecslip/in_1'],...
                    'position',[165,30,185,50])
add_line([sys,'/','ecslip'],[150,70;210,70])
add_line([sys,'/','ecslip'],[245,80;280,80])
add_line([sys,'/','ecslip'],[355,80;375,80])
add_line([sys,'/','ecslip'],[190,40;190,90;210,90])

%    Finished composite block 'ecslip'.

set_param([sys,'/','ecslip'],...
                    'position',[450,(3.5)+32.5*(nr-1),480,53.5+32.5*(nr-1)])

add_block('built-in/Outport',[sys,'/','Out_slip'])
set_param([sys,'/','Out_slip'],...
                    'Port',nr+3,...
                    'position',[500,18.5+32.5*(nr-1),520,38.5+32.5*(nr-1)])


add_block('built-in/Outport',[sys,'/','Out_speed'])
set_param([sys,'/','Out_speed'],....
                    'Port',nr+4,...
                    'position',[420,50.5+32.5*(nr-1),440,70.5+32.5*(nr-1)])

add_line(sys,[400,59.5+32.5*(nr-1);415,59.5+32.5*(nr-1)])

add_line(sys,[485,24.5+32.5*(nr-1);495,24.5+32.5*(nr-1)])

add_line(sys,[385,25.5+32.5*(nr-1);445,25.5+32.5*(nr-1)])

for cont=1:nr

        pozi=cont;

        if cont==13
                cont=101;
        end
        if cont==47
                cont=102;
        end

%    Subsystem  'newindmot'.

new_system([sys,'/','newindmot',cont])
set_param([sys,'/','newindmot',cont],'Location',[-15,4194420,712,4195282])

%    Subsystem  'newindmot',cont,'/eltrq'.

new_system([sys,'/','newindmot',cont,'/eltrq'])
set_param([sys,'/','newindmot',cont,'/eltrq'],'Location',[365,362,891,750])

add_block('built-in/Sum',[sys,'/','newindmot',cont,'/eltrq/Sum5'])
set_param([sys,'/','newindmot',cont,'/eltrq/Sum5'],...
                    'orientation',2,...
                    'Font Size',12,...
                    'inputs','+-',...
                    'position',[250,157,270,208])

add_block('built-in/Outport',[sys,'/','newindmot',cont,'/eltrq/out_1'])
set_param([sys,'/','newindmot',cont,'/eltrq/out_1'],...
                    'orientation',2,...
                    'position',[35,175,55,195])
```

```
add_block('built-in/Product',[sys,'/','newindmot',cont,'/eltrq/Product6'])
set_param([sys,'/','newindmot',cont,'/eltrq/Product6'],...
                    'orientation',2,...
                    'Font Size',12,...
                    'position',[315,101,345,139])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/eltrq/in_1'])
set_param([sys,'/','newindmot',cont,'/eltrq/in_1'],...
                    'orientation',2,...
                    'position',[390,80,410,100])

add_block('built-in/Product',[sys,'/','newindmot',cont,'/eltrq/Product7'])
set_param([sys,'/','newindmot',cont,'/eltrq/Product7'],...
                    'orientation',2,...
                    'Font Size',12,...
                    'position',[320,211,350,249])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/eltrq/in_3'])
set_param([sys,'/','newindmot',cont,'/eltrq/in_3'],...
                    'orientation',2,...
                    'Port','3',...
                    'position',[405,200,425,220])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/eltrq/in_4'])
set_param([sys,'/','newindmot',cont,'/eltrq/in_4'],...
                    'orientation',2,...
                    'Port','4',...
                    'position',[410,245,430,265])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/eltrq/in_2'])
set_param([sys,'/','newindmot',cont,'/eltrq/in_2'],...
                    'orientation',2,...
                    'Port','2',...
                    'position',[395,135,415,155])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/eltrq/Gain19'])
set_param([sys,'/','newindmot',cont,'/eltrq/Gain19'],...
                    'orientation',2,...
                    'Font Size',12,...
                    'Gain',pp,...
                    'position',[90,164,135,206])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/eltrq/Gain18'])
set_param([sys,'/','newindmot',cont,'/eltrq/Gain18'],...
                    'orientation',2,...
                    'Font Size',12,...
                    'Gain',M,...
                    'position',[160,164,205,206])

add_line([sys,'/','newindmot',cont,'/eltrq'],[155,185;140,185])
add_line([sys,'/','newindmot',cont,'/eltrq'],[245,185;210,185])
add_line([sys,'/','newindmot',cont,'/eltrq'],[315,230;295,230;295,195;275,195])
add_line([sys,'/','newindmot',cont,'/eltrq'],[310,120;295,120;295,170;275,170])
add_line([sys,'/','newindmot',cont,'/eltrq'],[385,90;380,90;380,110;350,110])
add_line([sys,'/','newindmot',cont,'/eltrq'],[390,145;375,145;375,130;350,130])
add_line([sys,'/','newindmot',cont,'/eltrq'],[400,210;395,210;395,220;355,220])
add_line([sys,'/','newindmot',cont,'/eltrq'],[85,185;60,185])
add_line([sys,'/','newindmot',cont,'/eltrq'],[405,255;385,255;385,240;355,240])

%    Finished composite block 'newindmot',cont,'/eltrq'.

set_param([sys,'/','newindmot',cont,'/eltrq'],...
                    'position',[595,395,655,485])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/did'])
set_param([sys,'/','newindmot',cont,'/did'],...
```

```
                    'Font Size',12,...
                    'position',[305,505,310,510])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/iD'])
set_param([sys,'/','newindmot',cont,'/iD'],...
                    'Font Size',12,...
                    'position',[275,85,280,90])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/vD'])
set_param([sys,'/','newindmot',cont,'/vD'],...
                    'Font Size',12,...
                    'position',[455,60,460,65])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/diD'])
set_param([sys,'/','newindmot',cont,'/diD'],...
                    'Font Size',12,...
                    'position',[275,115,280,120])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/diQ'])
set_param([sys,'/','newindmot',cont,'/diQ'],...
                    'Font Size',12,...
                    'position',[270,280,275,285])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/id'])
set_param([sys,'/','newindmot',cont,'/id'],...
                    'Font Size',12,...
                    'position',[295,565,300,570])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/iq'])
set_param([sys,'/','newindmot',cont,'/iq'],...
                    'Font Size',12,...
                    'position',[295,685,300,690])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/diq'])
set_param([sys,'/','newindmot',cont,'/diq'],...
                    'Font Size',12,...
                    'position',[300,750,305,755])

%    Subsystem  'newindmot',cont,'/EQ4'.

new_system([sys,'/','newindmot',cont,'/EQ4'])
set_param([sys,'/','newindmot',cont,'/EQ4'],'Location',[370,109,1124,663])

add_block('built-in/Product',[sys,'/','newindmot',cont,'/EQ4/Product3'])
set_param([sys,'/','newindmot',cont,'/EQ4/Product3'],...
                    'Font Size',12,...
                    'position',[150,303,180,327])

add_block('built-in/Outport',[sys,'/','newindmot',cont,'/EQ4/out_2'])
set_param([sys,'/','newindmot',cont,'/EQ4/out_2'],...
                    'Port','2',...
                    'position',[155,15,175,35])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/EQ4/Gain11'])
set_param([sys,'/','newindmot',cont,'/EQ4/Gain11'],...
                    'Font Size',12,...
                    'Gain',1/Lq,...
                    'position',[45,90,95,140])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/EQ4/in_2'])
set_param([sys,'/','newindmot',cont,'/EQ4/in_2'],...
                    'Port','2',...
                    'position',[225,170,240,190])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/EQ4/diQ'])
set_param([sys,'/','newindmot',cont,'/EQ4/diQ'],...
                    'Font Size',12,...
```

```
                    'position',[270,160,275,165])

add_block('built-in/Integrator',[sys,'/','newindmot',cont,'/EQ4/Integrator3'])
set_param([sys,'/','newindmot',cont,'/EQ4/Integrator3'],...
                    'Font Size',12,...
                    'position',[175,94,205,136])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/EQ4/in_1'])
set_param([sys,'/','newindmot',cont,'/EQ4/in_1'],...
                    'orientation',2,...
                    'position',[505,15,525,35])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/EQ4/Gain12'])
set_param([sys,'/','newindmot',cont,'/EQ4/Gain12'],...
                    'Font Size',12,...
                    'Gain',Rq,...
                    'position',[330,98,370,132])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/EQ4/Gain15'])
set_param([sys,'/','newindmot',cont,'/EQ4/Gain15'],...
                    'Font Size',12,...
                    'Gain',MqQ,...
                    'position',[335,160,390,200])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/EQ4/Gain13'])
set_param([sys,'/','newindmot',cont,'/EQ4/Gain13'],...
                    'Font Size',12,...
                    'Gain',GqD,...
                    'position',[260,209,310,251])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/EQ4/Gain14'])
set_param([sys,'/','newindmot',cont,'/EQ4/Gain14'],...
                    'Font Size',12,...
                    'Gain',Gqd,...
                    'position',[265,294,315,336])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/EQ4/in_3'])
set_param([sys,'/','newindmot',cont,'/EQ4/in_3'],...
                    'Port','3',...
                    'position',[40,215,60,235])

add_block('built-in/Product',[sys,'/','newindmot',cont,'/EQ4/Product2'])
set_param([sys,'/','newindmot',cont,'/EQ4/Product2'],...
                    'Font Size',12,...
                    'position',[190,218,220,242])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/EQ4/iD'])
set_param([sys,'/','newindmot',cont,'/EQ4/iD'],...
                    'Font Size',12,...
                    'position',[135,200,140,205])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/EQ4/in_4'])
set_param([sys,'/','newindmot',cont,'/EQ4/in_4'],...
                    'Port','4',...
                    'position',[65,380,85,400])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/EQ4/id'])
set_param([sys,'/','newindmot',cont,'/EQ4/id'],...
                    'Font Size',12,...
                    'position',[110,280,115,285])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/EQ4/in_5'])
set_param([sys,'/','newindmot',cont,'/EQ4/in_5'],...
                    'Port','5',...
                    'position',[40,300,60,320])

add_block('built-in/Sum',[sys,'/','newindmot',cont,'/EQ4/Sum3'])
```

```
set_param([sys,'/','newindmot',cont,'/EQ4/Sum3'],...
                    'Font Size',12,...
                    'inputs','+----',...
                    'position',[470,84,515,186])

add_block('built-in/Outport',[sys,'/','newindmot',cont,'/EQ4/out_1'])
set_param([sys,'/','newindmot',cont,'/EQ4/out_1'],...
                    'position',[320,20,340,40])
add_line([sys,'/','newindmot',cont,'/EQ4'],[320,315;455,315;465,175])
add_line([sys,'/','newindmot',cont,'/EQ4'],[315,230;445,230;445,155;465,155])
add_line([sys,'/','newindmot',cont,'/EQ4'],[185,315;260,315])
add_line([sys,'/','newindmot',cont,'/EQ4'],[225,230;255,230])
add_line([sys,'/','newindmot',cont,'/EQ4'],[395,180;440,180;440,135;465,135])
add_line([sys,'/','newindmot',cont,'/EQ4'],[100,115;170,115])
add_line([sys,'/','newindmot',cont,'/EQ4'],[375,115;465,115])
add_line([sys,'/','newindmot',cont,'/EQ4'],[210,115;325,115])
add_line([sys,'/','newindmot',cont,'/EQ4'],[500,25;445,25;445,95;465,95])
add_line([sys,'/','newindmot',cont,'/EQ4'],[100,115;120,115;120,25;150,25])
add_line([sys,'/','newindmot',cont,'/EQ4'],[245,180;330,180])
add_line([sys,'/','newindmot',cont,'/EQ4'],[65,225;185,225])
add_line([sys,'/','newindmot',cont,'/EQ4'],[65,310;145,310])
add_line([sys,'/','newindmot',cont,'/EQ4'],[90,390;90,235;185,235])
add_line([sys,'/','newindmot',cont,'/EQ4'],[90,320;145,320])
add_line([sys,'/','newindmot',cont,'/EQ4'],[295,115;295,30;315,30])
add_line([sys,'/','newindmot',cont,'/EQ4'],[520,135;555,135;555,66;20,65;20,115;40,115])

%    Finished composite block 'newindmot',cont,'/EQ4'.

set_param([sys,'/','newindmot',cont,'/EQ4'],...
                    'position',[210,672,280,798])

%    Subsystem  'newindmot',cont,'/EQ3'.
new_system([sys,'/','newindmot',cont,'/EQ3'])
set_param([sys,'/','newindmot',cont,'/EQ3'],'Location',[430,176,1063,764])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/EQ3/Gain9'])
set_param([sys,'/','newindmot',cont,'/EQ3/Gain9'],...
                    'Font Size',12,...
                    'Gain',GdQ,...
                    'position',[320,210,370,260])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/EQ3/Gain10'])
set_param([sys,'/','newindmot',cont,'/EQ3/Gain10'],...
                    'Font Size',12,...
                    'Gain',Gdq,...
                    'position',[315,290,365,340])

add_block('built-in/Integrator',[sys,'/','newindmot',cont,'/EQ3/Integrator2'])
set_param([sys,'/','newindmot',cont,'/EQ3/Integrator2'],...
                    'Font Size',12,...
                    'position',[165,76,190,114])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/EQ3/Gain7'])
set_param([sys,'/','newindmot',cont,'/EQ3/Gain7'],...
                    'Font Size',12,...
                    'Gain',Rd,...
                    'position',[335,76,380,114])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/EQ3/Gain8'])
set_param([sys,'/','newindmot',cont,'/EQ3/Gain8'],...
                    'Font Size',12,...
                    'Gain',MdD,...
                    'position',[325,135,375,185])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/EQ3/Gain6'])
set_param([sys,'/','newindmot',cont,'/EQ3/Gain6'],...
```

```
                    'Font Size',12,...
                    'Gain',1/Ld,...
                    'position',[60,70,110,120])

add_block('built-in/Sum',[sys,'/','newindmot',cont,'/EQ3/Sum2'])
set_param([sys,'/','newindmot',cont,'/EQ3/Sum2'],...
                    'Font Size',12,...
                    'inputs','+----',...
                    'position',[430,72,460,148])

add_block('built-in/Outport',[sys,'/','newindmot',cont,'/EQ3/out_1'])
set_param([sys,'/','newindmot',cont,'/EQ3/out_1'],...
                    'position',[160,15,180,35])

add_block('built-in/Outport',[sys,'/','newindmot',cont,'/EQ3/out_2'])
set_param([sys,'/','newindmot',cont,'/EQ3/out_2'],...
                    'Port','2',...
                    'position',[315,10,335,30])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/EQ3/in_1'])
set_param([sys,'/','newindmot',cont,'/EQ3/in_1'],...
                    'orientation',2,...
                    'position',[480,35,500,55])
add_block('built-in/Note',[sys,'/','newindmot',cont,'/EQ3/diD'])
set_param([sys,'/','newindmot',cont,'/EQ3/diD'],...
                    'Font Size',12,...
                    'position',[280,135,285,140])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/EQ3/in_2'])
set_param([sys,'/','newindmot',cont,'/EQ3/in_2'],...
                    'Port','2',...
                    'position',[230,150,250,170])

add_block('built-in/Product',[sys,'/','newindmot',cont,'/EQ3/Product'])
set_param([sys,'/','newindmot',cont,'/EQ3/Product'],...
                    'Font Size',12,...
                    'position',[230,223,260,247])

add_block('built-in/Product',[sys,'/','newindmot',cont,'/EQ3/Product1'])
set_param([sys,'/','newindmot',cont,'/EQ3/Product1'],...
                    'Font Size',12,...
                    'position',[235,303,265,327])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/EQ3/iQ'])
set_param([sys,'/','newindmot',cont,'/EQ3/iQ'],...
                    'Font Size',12,...
                    'position',[160,205,165,210])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/EQ3/iq'])
set_param([sys,'/','newindmot',cont,'/EQ3/iq'],...
                    'Font Size',12,...
                    'position',[180,285,185,290])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/EQ3/in_3'])
set_param([sys,'/','newindmot',cont,'/EQ3/in_3'],...
                    'Port','3',...
                    'position',[60,220,80,240])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/EQ3/in_5'])
set_param([sys,'/','newindmot',cont,'/EQ3/in_5'],...
                    'Port','5',...
                    'position',[60,300,80,320])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/EQ3/in_4'])
set_param([sys,'/','newindmot',cont,'/EQ3/in_4'],...
                    'Port','4',...
                    'position',[60,350,80,370])
```

```
add_line([sys,'/','newindmot',cont,'/EQ3'],[195,95;330,95])
add_line([sys,'/','newindmot',cont,'/EQ3'],[270,315;310,315])
add_line([sys,'/','newindmot',cont,'/EQ3'],[265,235;315,235])
add_line([sys,'/','newindmot',cont,'/EQ3'],[370,315;415,315;425,140])
add_line([sys,'/','newindmot',cont,'/EQ3'],[375,235;405,235;405,125;425,125])
add_line([sys,'/','newindmot',cont,'/EQ3'],[380,160;395,160;395,110;425,110])
add_line([sys,'/','newindmot',cont,'/EQ3'],[465,110;465,65;45,65;55,95])
add_line([sys,'/','newindmot',cont,'/EQ3'],[385,95;425,95])
add_line([sys,'/','newindmot',cont,'/EQ3'],[115,95;160,95])
add_line([sys,'/','newindmot',cont,'/EQ3'],[475,45;410,45;410,80;425,80])
add_line([sys,'/','newindmot',cont,'/EQ3'],[115,95;145,95;155,25])
add_line([sys,'/','newindmot',cont,'/EQ3'],[195,95;300,95;310,20])
add_line([sys,'/','newindmot',cont,'/EQ3'],[255,160;320,160])
add_line([sys,'/','newindmot',cont,'/EQ3'],[85,230;225,230])
add_line([sys,'/','newindmot',cont,'/EQ3'],[85,360;150,360;150,240;225,240])
add_line([sys,'/','newindmot',cont,'/EQ3'],[85,310;230,310])
add_line([sys,'/','newindmot',cont,'/EQ3'],[85,360;150,360;150,320;230,320])

%    Finished composite block 'newindmot',cont,'/EQ3'.

set_param([sys,'/','newindmot',cont,'/EQ3'],...
                    'position',[205,499,275,621])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/vq'])
set_param([sys,'/','newindmot',cont,'/vq'],...
                    'Font Size',12,...
                    'position',[400,610,405,615])

add_block('built-in/Constant',[sys,'/','newindmot',cont,'/Constant2'])
set_param([sys,'/','newindmot',cont,'/Constant2'],...
                    'orientation',2,...
                    'Value','0',...
                    'position',[420,625,440,645])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/vd'])
set_param([sys,'/','newindmot',cont,'/vd'],...
                    'Font Size',12,...
                    'position',[395,450,400,455])

add_block('built-in/Constant',[sys,'/','newindmot',cont,'/Constant1'])
set_param([sys,'/','newindmot',cont,'/Constant1'],...
                    'orientation',2,...
                    'Value','0',...
                    'position',[420,460,440,480])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/iQ'])
set_param([sys,'/','newindmot',cont,'/iQ'],...
                    'Font Size',12,...
                    'position',[275,245,280,250])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/vQ'])
set_param([sys,'/','newindmot',cont,'/vQ'],...
                    'Font Size',12,...
                    'position',[470,195,475,200])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/pp'])
set_param([sys,'/','newindmot',cont,'/pp'],...
                    'orientation',3,...
                    'Font Size',12,...
                    'Gain',pp,...
                    'position',[79,765,121,810])

%    Subsystem  'newindmot',cont,'/EQ2'.

new_system([sys,'/','newindmot',cont,'/EQ2'])
set_param([sys,'/','newindmot',cont,'/EQ2'],'Location',[439,232,960,444])
```

```
add_block('built-in/Gain',[sys,'/','newindmot',cont,'/EQ2/Gain5'])
set_param([sys,'/','newindmot',cont,'/EQ2/Gain5'],...
                    'Font Size',12,...
                    'Gain',MQq,...
                    'position',[305,140,355,190])

add_block('built-in/Integrator',[sys,'/','newindmot',cont,'/EQ2/Integrator1'])
set_param([sys,'/','newindmot',cont,'/EQ2/Integrator1'],...
                    'Font Size',12,...
                    'position',[190,76,215,114])

add_block('built-in/Sum',[sys,'/','newindmot',cont,'/EQ2/Sum1'])
set_param([sys,'/','newindmot',cont,'/EQ2/Sum1'],...
                    'Font Size',12,...
                    'inputs','+--',...
                    'position',[455,70,480,120])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/EQ2/Gain3'])
set_param([sys,'/','newindmot',cont,'/EQ2/Gain3'],...
                    'Font Size',12,...
                    'Gain',1/LQ,...
                    'position',[70,70,120,120])

add_block('built-in/Outport',[sys,'/','newindmot',cont,'/EQ2/out_2'])
set_param([sys,'/','newindmot',cont,'/EQ2/out_2'],...
                    'Port','2',...
                    'position',[150,10,170,30])
add_block('built-in/Outport',[sys,'/','newindmot',cont,'/EQ2/out_1'])
set_param([sys,'/','newindmot',cont,'/EQ2/out_1'],...
                    'position',[245,10,265,30])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/EQ2/in_2'])
set_param([sys,'/','newindmot',cont,'/EQ2/in_2'],...
                    'Port','2',...
                    'position',[230,155,250,175])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/EQ2/diq'])
set_param([sys,'/','newindmot',cont,'/EQ2/diq'],...
                    'Font Size',12,...
                    'position',[275,140,280,145])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/EQ2/in_1'])
set_param([sys,'/','newindmot',cont,'/EQ2/in_1'],...
                    'position',[395,10,415,30])
add_block('built-in/Gain',[sys,'/','newindmot',cont,'/EQ2/Gain4'])
set_param([sys,'/','newindmot',cont,'/EQ2/Gain4'],...
                    'Font Size',12,...
                    'Gain',RQ,...
                    'position',[300,73,350,117])
add_line([sys,'/','newindmot',cont,'/EQ2'],[220,95;295,95])
add_line([sys,'/','newindmot',cont,'/EQ2'],[485,95;505,95;505,65;55,65;65,95])
add_line([sys,'/','newindmot',cont,'/EQ2'],[360,165;390,165;390,110;450,110])
add_line([sys,'/','newindmot',cont,'/EQ2'],[355,95;450,95])
add_line([sys,'/','newindmot',cont,'/EQ2'],[125,95;185,95])
add_line([sys,'/','newindmot',cont,'/EQ2'],[420,20;430,20;430,80;450,80])
add_line([sys,'/','newindmot',cont,'/EQ2'],[220,95;225,95;225,20;240,20])
add_line([sys,'/','newindmot',cont,'/EQ2'],[125,95;130,95;130,20;145,20])
add_line([sys,'/','newindmot',cont,'/EQ2'],[255,165;300,165])
%    Finished composite block 'newindmot',cont,'/EQ2'.
set_param([sys,'/','newindmot',cont,'/EQ2'],...
                    'position',[210,254,260,316])
%    Subsystem  'newindmot',cont,'/EQ1'.
new_system([sys,'/','newindmot',cont,'/EQ1'])
set_param([sys,'/','newindmot',cont,'/EQ1'],'Location',[217,186,830,595])

add_block('built-in/Integrator',[sys,'/','newindmot',cont,'/EQ1/Integrator'])
set_param([sys,'/','newindmot',cont,'/EQ1/Integrator'],...
```

```
                    'Font Size',12,...
                    'position',[195,91,220,129])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/EQ1/Gain1'])
set_param([sys,'/','newindmot',cont,'/EQ1/Gain1'],...
                    'Font Size',12,...
                    'Gain',RD,...
                    'position',[340,91,385,129])

add_block('built-in/Sum',[sys,'/','newindmot',cont,'/EQ1/Sum'])
set_param([sys,'/','newindmot',cont,'/EQ1/Sum'],...
                    'Font Size',12,...
                    'inputs','+--',...
                    'position',[445,82,470,138])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/EQ1/Gain'])
set_param([sys,'/','newindmot',cont,'/EQ1/Gain'],...
                    'Font Size',12,...
                    'Gain',1/LD,...
                    'position',[75,85,125,135])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/EQ1/in_1'])
set_param([sys,'/','newindmot',cont,'/EQ1/in_1'],...
                    'position',[390,25,410,45])

add_block('built-in/Outport',[sys,'/','newindmot',cont,'/EQ1/out_1'])
set_param([sys,'/','newindmot',cont,'/EQ1/out_1'],...
                    'position',[265,15,285,35])

add_block('built-in/Outport',[sys,'/','newindmot',cont,'/EQ1/out_2'])
set_param([sys,'/','newindmot',cont,'/EQ1/out_2'],...
                    'Port','2',...
                    'position',[145,20,165,40])

add_block('built-in/Gain',[sys,'/','newindmot',cont,'/EQ1/Gain2'])
set_param([sys,'/','newindmot',cont,'/EQ1/Gain2'],...
                    'Font Size',12,...
                    'Gain',MDd,...
                    'position',[345,155,395,205])

add_block('built-in/Note',[sys,'/','newindmot',cont,'/EQ1/did'])
set_param([sys,'/','newindmot',cont,'/EQ1/did'],...
                    'Font Size',12,...
                    'position',[300,155,305,160])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/EQ1/in_2'])
set_param([sys,'/','newindmot',cont,'/EQ1/in_2'],...
                    'Port','2',...
                    'position',[245,170,265,190])
add_line([sys,'/','newindmot',cont,'/EQ1'],[475,110;475,75;55,75;55,110;70,110])
add_line([sys,'/','newindmot',cont,'/EQ1'],[400,180;430,180;440,130])
add_line([sys,'/','newindmot',cont,'/EQ1'],[390,110;440,110])
add_line([sys,'/','newindmot',cont,'/EQ1'],[225,110;335,110])
add_line([sys,'/','newindmot',cont,'/EQ1'],[130,110;190,110])
add_line([sys,'/','newindmot',cont,'/EQ1'],[415,35;420,35;420,90;440,90])
add_line([sys,'/','newindmot',cont,'/EQ1'],[225,110;235,110;235,25;260,25])
add_line([sys,'/','newindmot',cont,'/EQ1'],[130,110;140,30])
add_line([sys,'/','newindmot',cont,'/EQ1'],[270,180;340,180])

%    Finished composite block 'newindmot',cont,'/EQ1'.

set_param([sys,'/','newindmot',cont,'/EQ1'],...
                    'position',[210,91,255,149])

add_block('built-in/Inport',[sys,'/','newindmot',cont,'/in_1'])
set_param([sys,'/','newindmot',cont,'/in_1'],...
                    'position',[610,75,630,95])
```

```
add_block('built-in/Inport',[sys,'/','newindmot',cont,'/in_2'])
set_param([sys,'/','newindmot',cont,'/in_2'],...
                    'Port','2',...
                    'position',[610,190,630,210])
add_block('built-in/Outport',[sys,'/','newindmot',cont,'/out_1'])
set_param([sys,'/','newindmot',cont,'/out_1'],...
                    'position',[685,430,705,450])
add_block('built-in/Inport',[sys,'/','newindmot',cont,'/in_3'])
set_param([sys,'/','newindmot',cont,'/in_3'],...
                    'Port','3',...
                    'position',[610,820,630,840])
add_line([sys,'/','newindmot',cont],[100,760;45,760;45,585;200,585])
add_line([sys,'/','newindmot',cont],[260,135;305,135;305,195;65,195;65,535;200,535])
add_line([sys,'/','newindmot',cont],[265,300;355,300;355,385;100,385;100,710;205,710])
add_line([sys,'/','newindmot',cont],[280,530;330,530;330,445;155,445;155,135;205,135])
add_line([sys,'/','newindmot',cont],[285,705;326,705;326,655;145,655;145,610;200,610])
add_line([sys,'/','newindmot',cont],[325,705;538,705;538,430;590,430])
add_line([sys,'/','newindmot',cont],[285,770;370,770;370,345;115,345;115,300;205,300])
add_line([sys,'/','newindmot',cont],[260,105;321,105;321,170;86,170;86,735;205,735])
add_line([sys,'/','newindmot',cont],[321,170;325,170;325,410;590,410])
add_line([sys,'/','newindmot',cont],[280,590;345,590;345,820;175,820;175,785;205,785])
add_line([sys,'/','newindmot',cont],[345,760;485,760;485,470;590,470])
add_line([sys,'/','newindmot',cont],[265,270;290,270;290,370;135,370;135,560;200,560])
add_line([sys,'/','newindmot',cont],[290,270;290,265;476,265;477,305;477,450;590,450])
add_line([sys,'/','newindmot',cont],[415,470;111,470;111,510;200,510])
add_line([sys,'/','newindmot',cont],[415,635;116,635;116,685;205,685])
add_line([sys,'/','newindmot',cont],[100,760;205,760])
add_line([sys,'/','newindmot',cont],[635,85;175,85;175,105;205,105])
add_line([sys,'/','newindmot',cont],[635,200;170,200;170,270;205,270])
add_line([sys,'/','newindmot',cont],[635,830;100,830;100,815])
add_line([sys,'/','newindmot',cont],[660,440;680,440])

%    Finished composite block 'newindmot'.
set_param([sys,'/','newindmot',cont],...
                    'position',[105,12+65*(pozi-1),135,64+65*(pozi-1)])
amp=newamp(pozi);
frq=newfq(pozi);
ph=newphase(pozi);
add_block('built-in/Sine Wave',[sys,'/','Sine1',cont])
set_param([sys,'/','Sine1',cont],...
                    'amplitude',amp,...
                    'frequency',frq,...
                    'phase',ph+pi,...
                    'position',[15,28+65*(pozi-1),35,48+65*(pozi-1)])
add_block('built-in/Sine Wave',[sys,'/','Sine',cont])
set_param([sys,'/','Sine',cont],...
                    'amplitude',amp,...
                    'frequency',frq,...
                    'phase',ph+pi/2,...
                    'position',[40,13+65*(pozi-1),60,33+65*(pozi-1)])

add_line(sys,[65,22+65*(pozi-1);100,22+65*(pozi-1)])
add_line(sys,[40,41+65*(pozi-1);100,41+65*(pozi-1)])
add_line(sys,[85,54+65*(pozi-1);100,54+65*(pozi-1)])

add_block('built-in/Outport',[sys,'/','Outport',cont])
set_param([sys,'/','Outport',cont],...
                    'Port',pozi,...
                    'position',[215,11+65*(pozi-1),235,31+65*(pozi-1)])
add_line(sys,[140,38+65*(pozi-1);265,38+65*(pozi-1)])
add_line(sys,[140,35+65*(pozi-1);190,35+65*(pozi-1);190,20+65*(pozi-1);210,20+65*(pozi-1)])
drawnow
end
add_block('built-in/Clock',[sys,'/','Clock'])
set_param([sys,'/','Clock'],...
                    'position',[550,5,570,25])
```

```
add_block('built-in/Outport',[sys,'/','time'])
set_param([sys,'/','time'],...
                    'Port',nr+2,...
                    'position',[590,5,610,25])

add_line(sys,[575,15;585,15])

answer=input('would you like to run the simulation? y/n..','s');
a='y';
rz=strcmp(a,answer);
if rz==1
        tf=input('for how long you want to run the simulation?  ')
        disp('Now we are running the simmulation ,you will have to wait a bit..')
        [t,x,y]=rk45('mmprognew',tf);
        disp('so , these are the plots..')
        timp=y(:,nr+2);
        for cont=1:nr
                figure
                plot(timp,y(:,cont),'k')
                xlabel('TIME')
                ylabel('TORQUE')
        end
        for cont=1:nr
                figure
                plot(y(:,nr+4),y(:,cont),'k')
                xlabel('SPEED')
                ylabel('TORQUE')
                grid
        end
        figure
        plot(timp,y(:,nr+1),'k')
        xlabel('TIME')
        ylabel('final TORQUE')
        figure
        plot(timp,y(:,nr+3),'k')
        xlabel('TIME')
        ylabel('SLIP')
        figure
        plot(timp,y(:,nr+4),'k')
        xlabel('TIME')
        ylabel('SPEED')
end
% Return any arguments.
if (nargin | nargout)
        % Must use feval here to access system in memory
        if (nargin > 3)
                if (flag == 0)
                        eval(['[ret,x0,str,ts,xts]=',sys,'(t,x,u,flag);'])
                else
                        eval(['ret =', sys,'(t,x,u,flag);'])
                end
        else
                [ret,x0,str,ts,xts] = feval(sys);
        end
else
        drawnow % Flash up the model and execute load callback

end
```

APPENDIX 3 - *The procedures implemented in assembler language for the microcontroller SAB 80C166*

1. The total leakage reactance estimation: *the listing is part of the program calc.uno.*

```
;********* constants inserted on  24.09.1997**********************

cpoz      EQU     0CCCh           ; current limit=0.8
;cvd      EQU     1232d           ;1/3*uz in 4.12
;cvdneg   EQU     0FB30h          ;64816d  (-1/3)*uz in 4.12
cvd       EQU     995h            ;  2/3*uz
cvdneg    EQU     0F66Bh          ; -2/3*uz
;tau      EQU     002Ah           ; 42 steps=0.009/213.33, integer
tau       EQU     002Dh           ; 45 steps=0.009/200, integer
;tau      EQU     0015h           ; 21 steps for Tc=426.66 ,integer
;tcw      EQU     0225h           ;(4.12)---> 314.16*426.66us=wb*Tc
tcw       EQU     0101h           ;(4.12)---> 314.16*200us  =wb*Tc

;********this part was added today, 24.09.1997*****
;*****************************************************

 MOV R1, ucmm       ;controlling the starting moment (when the procedure will be started..)
 MOV R2, #0
 ADD R2,R1
 JMP cc_Z, niente3

 MOV R1, conta
 ADD R1, #1
 MOV conta, R1

 MOV R7, salv7
 JB R7.0, tenspoz
 MOV R1, #cvd          ; v_beta is always 0
 MOV Vas12, R1         ; v_alfa is imposed 1/3*ud
 BSET R7.0
 MOV salv7, R7

tenspoz:
        MOV R7, salv7
        JB R7.1, niente1
        MOV R2, #cpoz         ;0.8 decimal
        MOV R1, i_alfa
        SUB R2, R1
        JMP cc_N, corto1
        JMP cc_UC, niente3
corto1:
        MOV R3, #0
        MOV Vas12, R3
        BSET R7.1
        MOV salv7, R7

niente1:
    MOV R7, salv7
    JB R7.2, niente2
    MOV R1, conta
    MOV R2, #tau              ;tau=0.009 sec
    SUB R2,R1
```

```
    JMP cc_N, tensneg
    JMP cc_UC, niente3
tensneg:
    MOV R3, #cvdneg
    MOV Vas12, R3
    MOV R1, i_alfa
    MOV i3, R1
    MOV R2, conta
    MOV temp3, R2
    BSET R7.2
    MOV salv7, R7
niente2:
    MOV R7, salv7
    JB R7.3, niente3
    MOV R1, i_alfa          ; i_alfa is negative
    MOV R2, #0
    SUB R2, R1              ; i_alfa becomes positive
    MOV R3, #cpoz           ; cneg=0.8
    SUB R3, R2
    JMP cc_N, corto2
    JMP cc_UC, niente3
corto2:
    MOV R3, #0
    MOV Vas12, R3
    MOV R1,i_alfa
    MOV i4, R1
    MOV R2, conta
    MOV temp4, R2
    BSET R7.3
    MOV salv7, R7
calcolo:
        MOV   R5, temp3     ;pay attention ! temp3 , temp4 are integers
        MOV   R6, temp4
        SUB   R6, R5
        MOV   R3, #cvd      ;R3<--- 1/3*uz or 2/3*uz
        MOV   R4, #tcw
        MUL   R4, R3
        CALL NORMAL2        ;R2 <-----1/3*uz*Tc*w  (4.12)
        MUL   R6, R2        ;R6 contains an integer, R2 a number in 4.12
        MOV   R6, MDL       ;R6 <-----1/3*uz*Tc*w*(temp4-temp3)
        MOV   nsop, R6

        MOV   R3, i3
        MOV   R4, i4
        SUB   R3, R4            ; R3 <---(i3-i4)
        MOV   R1, #3E8h
        MUL   R3, R1

        MOVBZ  R1, MDLH
        SHR   R1, #4
        MOV   R2, MDH
        SHL   R2, #4
        OR    R2, R1
        MOV   nsot, R2      ;an integer (~?988?)

        MOV   R1, nsop
        MOV   R3, #3E8h     ;R3 <--- 1000 , integer
        MUL   R3, R1        ;1000*(u_cfin-usalm_a) ----->the result in MD
        MOV   R4, nsot
        DIVL  R4            ;MDL <--- MD/ R4
        MOV   R2, MDL
        MOV   x_sig, R2

;       division R2/R3 that is (1/3*uz*Tc*w*nrsteps)/(i3-i4)
; once this is calculated , the leakage reactance is obtained in pu**** x_sig****
niente3:
```

```
;*******************************
;the end of insertion 24.09.1997
```

2. The stator resistance estimation:

```
;******** constants added on 8/10/1997***************
;i_a      EQU   0400h        ;0.25 pu in 4.12 format
;i_b      EQU   0FC00h       ;-0.25 pu
;i_a      EQU   0333h        ; 0.2 pu in 4.12 format
;i_b      EQU   0FCCDh       ; -0.2 pu
i_a       EQU   0266h        ; 0.15 pu
i_b       EQU   0FD9Ah       ; -0.15 pu
;i_c      EQU   0800h        ;0.5 pu
;i_c      EQU   0E66h        ;0.9 pu
;i_c      EQU   1000h        ;1 pu
i_c       EQU   0CCDh        ;0.8 pu
i_d       EQU   0h

;************Tc=426.66 us*****************
;tau_a  EQU   1B78h      ;3sec/Tc     Tc=426.66
;tau_b  EQU   36EFh      ;6sec/Tc
;tau_c  EQU   5266h      ;9sec/Tc
;tau_a  EQU   1250h      ;2sec/Tc     Tc=426.66 us
;tau_b  EQU   249Fh      ;4sec/Tc
;tau_c  EQU   36EFh      ;6sec/Tc

;************Tc=200 us**********************
;tau_a  EQU    3A98h      ;3sec/Tc     Tc=200 us
;tau_b  EQU    7530h      ;6sec/Tc
;tau_c  EQU    0AFC8h     ;9sec/Tc
tau_a   EQU    2710h      ;2sec/Tc     Tc=200 us
tau_b   EQU    4E20h      ;4sec/Tc
tau_c   EQU    7530h      ;6sec/Tc
tau_cm  EQU    61A8h      ;5sec/Tc

;************this part added today , 8/10/1997***anca n.******
MOV    R1, ucmm
MOV    R2, #0h
ADD    R2, R1
JMP    cc_Z, next5

MOV    R1, conta
ADD    R1, #1h
MOV    conta, R1

MOV    R7, salv7
JB     R7.0, next1
MOV    R1, #i_a          ;0.25 pu
MOV    ialr, R1
BSET   R7.0
MOV    salv7, R7
JMP    cc_UC, next4      ; jumps at the end , before the regulators

next1:
        MOV    R7, salv7
        JB     R7.1, next2
        MOV    R1, conta
        MOV    R2, #tau_a     ;tau_a=3sec/Tc
        SUB    R2, R1
        JMP    cc_N, periodb
        JMP    cc_UC, calcoli_a
periodb:
        MOV    R1, #i_b          ; (-0.25) pu
        MOV    ialr, R1
```

```
        BSET   R7.1
        MOV    salv7, R7
next2:
        MOV    R7, salv7
        JB     R7.2, next3
        MOV    R1, conta
        MOV    R2, #tau_b     ;tau_b=6sec/Tc
        SUB    R2, R1
        JMP    cc_N, periodc
        JMP    cc_UC, next4
periodc:
        MOV    R1, #i_c              ; 1 pu
        MOV    ialr, R1
        BSET   R7.2
        MOV    salv7, R7
        JMP    cc_UC, next4
next3:
        MOV    R7, salv7
        JB     R7.3, next6
        MOV    R1, conta
        MOV    R2, #tau_c     ;tau_c=9sec/Tc
        SUB    R2, R1
        JMP    cc_N, periodd
        JMP    cc_UC, calcoli_c
periodd:
        MOV    R1, #i_d              ; 0 pu
        MOV    ialr, R1
        BSET   R7.3
        MOV    salv7, R7
        JMP    cc_UC, calcoli_fi
calcoli_a:
        MOV    R1, usalm_al
;       MOV    R2, Vas12
        MOV    R2, Uam
        ADD    R1, R2
        MOV    usalm_al, R1
        JMP    cc_NC, calc_i
        MOV    R3, usalm_ah
        ADD    R3, #1h
        MOV    usalm_ah, R3
    calc_i:
                MOV    R1, isalm_al
                MOV    R2, i_alfa
                ADD    R1, R2
                MOV    isalm_al, R1
                JMP    cc_NC, calc_conta
                MOV    R3, isalm_ah
                ADD    R3, #1h
                MOV    isalm_ah, R3
    calc_conta:
                MOV    R1, conta_a
                MOV    R2, #1h
                ADD    R1, R2
                MOV    conta_a, R1
next6:
        JMP    cc_UC, next4
calcoli_c:
        MOV    R1, conta
        MOV    R2, #tau_cm
        SUB    R1, R2
        JMP    cc_N, calc_ic

        MOV    R1, u_cfinl
;       MOV    R2, Vas12
        MOV    R2, Uam
        ADD    R1, R2
```

```
        MOV   u_cfinl, R1
        JMP   cc_NC, calc_c
        MOV   R3, u_cfinh
        ADD   R3, #1h
        MOV   u_cfinh, R3
    calc_c:
              MOV   R1, conta_cm
              MOV   R2, #1h
              ADD   R1, R2
              MOV   conta_cm, R1
    calc_ic:
              MOV   R1, isalm_cl
              MOV   R2, i_alfa
              ADD   R1, R2
              MOV   isalm_cl, R1
              JMP   cc_NC, calc_cont
              MOV   R3, isalm_ch
              ADD   R3, #1h
              MOV   isalm_ch, R3
    calc_cont:
              MOV   R1, conta_c
              MOV   R2, #1h
              ADD   R1, R2
              MOV   conta_c, R1
              JMP   cc_UC, next4
calcoli_fi:
        MOV   MDL, isalm_al      ;              isalm_a
        MOV   MDH, isalm_ah      ; isalm_a = ----------
        MOV   R2, conta_a        ;              conta_a
        DIVL  R2
        MOV   isalm_a, MDL   ;

        MOV   MDL, usalm_al
        MOV   MDH, usalm_ah
        MOV   R2, conta_a
        DIVL  R2
        MOV   usalm_a, MDL

        MOV   MDL, u_cfinl
        MOV   MDH, u_cfinh
        MOV   R2, conta_cm
        DIVL  R2
        MOV   u_cfin, MDL

        MOV   MDL, isalm_cl
        MOV   MDH, isalm_ch
        MOV   R2, conta_c
        DIVL  R2
        MOV   isalm_c, MDL
calcolo:
        MOV   R3, isalm_c
        MOV   R4, isalm_a
        SUB   R3, R4
        MOV   R1, #3E8h
        MUL   R3, R1

        MOVBZ  R1, MDLH
        SHR    R1, #4
        MOV    R2, MDH
        SHL    R2, #4
        OR     R2, R1
        MOV    nsot, R2              ;an integer (~550)

        MOV    R1, u_cfin
        MOV    R2, usalm_a
        SUB    R1, R2
```

```
        MOV    R3, #3E8h            ;R3 <--- 1000 , integer
        MUL    R3, R1               ;1000*(u_cfin-usalm_a)  -----> the result in MD
        MOV    R4, nsot
        DIVL   R4          ;MDL <--- MD/ R4
        MOV    R2, MDL
        MOV    r_s, R2
next4:
;       MOV    R1, salv7
;       MOV    R2, #v7
;       SUB    R1, R2
;       JMP    cc_Z, next5
; id and iq current regulators implementation follows..
;
next5:
```

3. The rotor resistance estimation:

```
;******** constants inserted on 8/10/1997***************

;i_a    EQU    0400h        ;0.25 pu in 4.12 format
;i_b    EQU    0FC00h       ;-0.25 pu
;i_a    EQU    0333h        ; 0.2 pu in 4.12 format
;i_b    EQU    0FCCDh       ; -0.2 pu
;i_a      EQU      0266h            ; 0.15 pu
;i_b      EQU      0FD9Ah           ; -0.15 pu
;i_a      EQU      0199h            ; 0.1 pu
;i_b      EQU      0FE67h           ; -0.1 pu
i_a       EQU      0CCh             ; 0.05 pu
i_b       EQU      0FF34h           ; -0.05 pu
;i_b      EQU      0266h            ; 0.15 pu
;i_c      EQU      0800h            ;0.5 pu
;i_c      EQU      1000h            ;1 pu
i_c       EQU      0CCDh            ;0.8 pu
;i_c      EQU      0E66h            ;0.9 pu
i_d       EQU      0h

;************Tc=426.66 us*****************
;tau_a    EQU    1B78h      ;3sec/Tc      Tc=426.66
;tau_b    EQU    36EFh      ;6sec/Tc
;tau_c    EQU    5266h      ;9sec/Tc
tau_a     EQU    1250h      ;2sec/Tc      Tc=426.66 us
tau_b     EQU    249Fh      ;4sec/Tc
tau_c     EQU    36EFh      ;6sec/Tc

;************Tc=200 us*********************
;tau_a    EQU    3A98h      ;3sec/Tc      Tc=200 us
;tau_b    EQU    7530h      ;6sec/Tc
;tau_c    EQU    0AFC8h     ;9sec/Tc
;tau_a    EQU    2710h      ;2sec/Tc      Tc=200 us
;tau_b    EQU    4E20h      ;4sec/Tc
;tau_c    EQU    7530h      ;6sec/Tc
rs_stim   EQU    1C4h       ;Tc=200 us, i_a=0.15 pu, i_b=-0.15 pu
                                  ; estimated using calc.due

;************inserimento**** 8/10/1997***anca n.******

MOV    R1, ucmm
MOV    R2, #0h
ADD    R2, R1
JMP    cc_Z, next5

MOV    R1, conta
ADD    R1, #1h
MOV    conta, R1
```

```
MOV    R7, salv7
JB     R7.0, next1
MOV    R1, #i_a            ;0.25 pu or 0.2 pu, 0.15 pu
MOV    ialr, R1
BSET   R7.0
MOV    salv7, R7
JMP    cc_UC, next4        ; jumps at the end , before the regulators

next1:
        MOV    R7, salv7
        JB     R7.1, next2
        MOV    R1, conta
        MOV    R2, #tau_a     ;tau_a=3sec/Tc
        SUB    R2, R1
        JMP    cc_N, periodb
        JMP    cc_UC, next4
periodb:
        MOV    R1, #i_b            ; (-0.25) pu or (-0.2), (-0.15)
        MOV    ialr, R1
        BSET   R7.1
        MOV    salv7, R7
        JMP    cc_UC, next4
next2:
        MOV    R7, salv7
        JB     R7.2, next3
        MOV    R1, conta
        MOV    R2, #tau_b     ;tau_b=6sec/Tc or 4sec/Tc
        SUB    R2, R1
        JMP    cc_N, periodc
        JMP    cc_UC, calcoli_b
periodc:
        MOV    R1, #i_c            ; 0.8 pu
        MOV    ialr, R1
        BSET   R7.2
        MOV    salv7, R7
        JMP    cc_UC, next4
next3:
        MOV    R7, salv7
        JB     R7.3, next6
        MOV    R1, conta
        MOV    R2, #tau_c     ;tau_c=9sec/Tc or 6sec/Tc
        SUB    R2, R1
        JMP    cc_N, periodd
        JMP    cc_UC, calcoli_c
periodd:
        MOV    R1, #i_d            ; 0 pu
        MOV    ialr, R1
        BSET   R7.3
        MOV    salv7, R7
        JMP    cc_UC, calcoli_fi
calcoli_b:
        MOV    R1, isalm_bl
        MOV    R2, i_alfa
        ADD    R1, R2
        MOV    isalm_bl, R1
        JMP    cc_NC, calc_conta
        MOV    R3, isalm_bh
        ADD    R3, #1h
        MOV    isalm_bh, R3
        calc_conta:

        MOV    R1, conta_b
        MOV    R2, #1h
        ADD    R1, R2
        MOV    conta_b, R1
next6:
```

```
        JMP    cc_UC, next4
calcoli_c:
;       MOV    R1, u_max
;       MOV    R2, Vas12
;       MOV    R2, Uam
;       SUB    R1, R2
;       JMP    cc_Z, camb
;       JMP    cc_N, camb
;    nocamb:
;    JMP    cc_UC, next4
;    camb:
;    MOV    u_max, R2
;    MOV    R1, i_alfa
;    MOV    i_umax, R1

              MOV    R1, i_umax
              MOV    R2, i_alfa
              SUB    R1, R2
              JMP    cc_NN, nocamb
             camb:
          MOV    i_umax, R2
          MOV    R1, Uam
          MOV    u_max, R1
          MOV    R2, conta
          MOV    conta_max, R2
          nocamb:
          JMP    cc_UC, next4
 calcoli_fi:
          MOV    MDL, isalm_bl      ;              isalm_b
          MOV    MDH, isalm_bh      ; isalm_b = ----------
          MOV    R2, conta_b        ;              conta_b
          DIVL   R2
          MOV    isalm_b, MDL   ;
calcolo:
          MOV    R4, isalm_b
          MOV    R3, i_umax
          SUB    R3, R4                ; R3 <--- i_umax+isalm_b
          MOV    R1, #3E8h             ;R1 <--- 1000 (integer)
          MUL    R3, R1

          MOVBZ  R1, MDLH
          SHR    R1, #4
          MOV    R2, MDH
          SHL    R2, #4
          OR     R2, R1
          MOV    nsot, R2              ;nsot <--- an integer (~???)

          MOV    R2, i_umax
          MOV    R3, #rs_stim
          MUL    R2, R3
          CALL NORMAL2          ; R2 <---  rs_stim*i_umax
          MOV    R1, u_max
          SUB    R1, R2
          MOV    R3, #3E8h             ;R3 <--- 1000 , integer
          MUL    R3, R1                ;1000*(u_max-rs_stim*i_umax)
                                       ;the result in MD
          MOV    R4, nsot
          DIVL   R4                    ;MDL <--- MD/ R4
          MOV    R2, MDL
          MOV    r_r, R2
; stator resistance =nsop/nsot
next4:

next5:
;the rest of the program follows...
```

4. The rotor time constant estimation:

```
;******** constants inserted on 8/10/1997***************

i_a      EQU   0400h      ;0.25 pu in 4.12 format
i_b      EQU   0FC00h     ;-0.25 pu
;i_a     EQU   0333h      ; 0.2 pu in formato 4.12
;i_b     EQU   0FCCDh     ; -0.2 pu
;i_a     EQU   0266h      ; 0.15 pu
;i_b     EQU   0FD9Ah     ; -0.15 pu
;i_a     EQU   0199h      ; 0.1 pu
;i_b     EQU   0FE67h     ; -0.1 pu
;i_a     EQU   0CCh       ; 0.05 pu
;i_b     EQU   0FF34h     ; -0.05 pu
;i_b     EQU   0266h      ; 0.15 pu
;i_c     EQU   0800h      ;0.5 pu
;i_c     EQU   0E66h      ;0.9 pu
;i_c     EQU   1000h      ;1 pu
i_c      EQU   0CCDh      ;0.8 pu
i_d      EQU   0h

;************Tc=426.66 us*****************
;tau_a    EQU   1B78h      ;3sec/Tc     Tc=426.66
;tau_b    EQU   36EFh      ;6sec/Tc
;tau_c    EQU   5266h      ;9sec/Tc
tau_a    EQU   1250h      ;2sec/Tc     Tc=426.66 us
tau_b    EQU   249Fh      ;4sec/Tc
tau_c    EQU   36EFh      ;6sec/Tc

;************Tc=200 us*********************
;tau_a    EQU   3A98h      ;3sec/Tc     Tc=200 us
;tau_b    EQU   7530h      ;6sec/Tc
;tau_c    EQU   0AFC8h     ;9sec/Tc
;tau_a    EQU   2710h      ;2sec/Tc     Tc=200 us
;tau_b    EQU   4E20h      ;4sec/Tc
;tau_c    EQU   7530h      ;6sec/Tc
ee       EQU   5E2h       ; 1/e=1/2.7183=0.3679
pas_c    EQU   4443h      ; 4.2666
;pas_c    EQU   2000h      ; 2
sei      EQU   2710h      ; integer 10^4
;repeat_lim      EQU   C350h  ; 10sec/200 us
repeat_lim       EQU   5B8Dh  ; 10 sec/426.66 us

;************this part was added today,  8/10/1997***anca n.******

MOV   R1, ucmm
MOV   R2, #0h
ADD   R2, R1
JMP   cc_Z, next5

MOV   R1, conta
ADD   R1, #1h
MOV   conta, R1

MOV   R7, salv7
JB    R7.0, next1
MOV   R1, #i_a                    ;0.25 pu or 0.2 pu, 0.15 pu
MOV   ialr, R1
BSET  R7.0
MOV   salv7, R7
```

```
JMP    cc_UC, next4          ;jumps at the end , before the regulators

next1:
        MOV    R7, salv7
        JB     R7.1, next2
        MOV    R1, conta
        MOV    R2, #tau_a          ;tau_a=3sec/Tc
        SUB    R2, R1
        JMP    cc_N, periodb
        JMP    cc_UC, next4
periodb:
        MOV    R1, #i_b            ; (-0.25) pu or (-0.2), (-0.15)
        MOV    ialr, R1
        BSET   R7.1
        MOV    salv7, R7
        JMP    cc_UC, next4
next2:
        MOV    R7, salv7
        JB     R7.2, next3
        MOV    R1, conta
        MOV    R2, #tau_b          ;tau_b=6sec/Tc or 4sec/Tc
        SUB    R2, R1
        JMP    cc_N, periodc
        JMP    cc_UC, calcoli_b
periodc:
        MOV    R1, #i_c            ; 0.8 pu
        MOV    ialr, R1
        BSET   R7.2
        MOV    salv7, R7
        JMP    cc_UC, next4
next3:
        MOV    R7, salv7
        JB     R7.3, next5
        MOV    R1, conta
        MOV    R2, #tau_c          ;tau_c=9sec/Tc or 6sec/Tc
        SUB    R2, R1
        JMP    cc_N, periodd
        JMP    cc_UC, next4
periodd:
        MOV    R1, #i_d            ; 0 pu
        MOV    ialr, R1
        BSET   R7.3
        MOV    salv7, R7
        JMP    cc_UC, calcoli_fi
calcoli_b:
    MOV    R2, Uam
    JB     R2.15, changes   ; if Uam is a negative number, jumps at `changes`
    JMP    cc_UC, next4
changes:
    MOV    R1, u_max
    SUB    R1, R2
    JMP    cc_N, camb_max
        no_camb:
                JMP    cc_UC, c_minim
        camb_max:
                MOV    u_max, R2
        c_minim:
                MOV    R1, Uam
                MOV    R2, u_min
                SUB    R1, R2
                JMP    cc_N, camb_min
                JMP    cc_UC, next4
                        camb_min:
                                MOV    u_min, R1
                                MOV    R2, conta
                                MOV    conta_min, R2
```

```
                                    JMP    cc_UC, next4
calcoli_fi:
          MOV    R1, u_max
          MOV    R2, u_min
          SUB    R2, R1                ; R2 <--- u_min-u_max
          MOV    R3, #ee               ;R3 <--- ee=1/exp(1)
          MUL    R3, R2
          CALL NORMAL2          ; R2 <--- (u_min-u_max)*(1/e)

          MOV    R1, u_max
          ADD    R1, R2
          MOV    u_67, R1
next5:

;************this part added on 8/10/1997***anca n.******

MOV    R1, #repeat_lim              ;  after 10 sec the procedure is repeated
MOV    R2, conta
SUB    R2, R1
JMP    cc_N, next4

MOV    R1, conta_2
ADD    R1, #1h
MOV    conta_2, R1

MOV    R6, salv6
JB     R6.0, next_a
MOV    R1, #i_a              ;0.25 pu or 0.2 pu, 0.15 pu
MOV    ialr, R1
BSET   R6.0
MOV    salv6, R6
JMP    cc_UC, next4          ; jumps at the end , before the regulators
next_a:
          MOV    R6, salv6
          JB     R6.1, next_b
          MOV    R1, conta_2
          MOV    R2, #tau_a            ;tau_a=3sec/Tc
          SUB    R2, R1
          JMP    cc_N, period_b
          JMP    cc_UC, next4
period_b:
          MOV    R1, #i_b              ; (-0.25) pu or (-0.2), (-0.15)
          MOV    ialr, R1
          BSET   R6.1
          MOV    salv6, R6
          JMP    cc_UC, next4
next_b:
          MOV    R6, salv6
          JB     R6.2, next_c
          MOV    R1, conta_2
          MOV    R2, #tau_b            ;tau_b=6sec/Tc or 4sec/Tc
          SUB    R2, R1
          JMP    cc_N, period_c
          JMP    cc_UC, calcoli_b2
period_c:
          MOV    R1, #i_c              ; 0.8 pu
          MOV    ialr, R1
          BSET   R6.2
          MOV    salv6, R6
          JMP    cc_UC, next4
next_c:
          MOV    R6, salv6
          JB     R6.3, next_f
          MOV    R1, conta_2
          MOV    R2, #tau_c            ;tau_c=9sec/Tc or 6sec/Tc
```

```
        SUB    R2, R1
        JMP    cc_N, period_d
        JMP    cc_UC, next4
period_d:
        MOV    R1, #i_d            ; 0 pu
        MOV    ialr, R1
        BSET   R6.3
        MOV    salv6, R6
        JMP    cc_UC, calcolo_tr
calcoli_b2:
        MOV    R1, u_67
        MOV    R2, Uam
        SUB    R2, R1
        JMP    cc_N, camb
          nocamb:
                    JMP    cc_UC, next4
          camb:
                   MOV     R2, Uam
                   MOV     u_67trov, R2
                   MOV     R2, conta_2
                   MOV     conta_67, R2
next_f:
        JMP    cc_UC, next4
calcolo_tr:
        MOV    R1, conta_67
        MOV    R2, conta_min
        SUB    R1, R2        ; R1 <--- conta_67-conta_min (integer)
        MOV    R3, #pas_c              ;R3 <--- step= Tc/100
        MUL    R3, R1
        MOV    R1, #sei
        DIVL   R1
        MOV    t_r, MDL
next4:
; here: current regulators implementation
next_e:
```

APPENDIX 4 - Motor Data sets

1. `LAFERT` motor

Vno	*rated voltage [rms]*	[V]	220
Ino	*rated current [rms]*	[A]	4.2
Fno	*rated frequency*	[Hz]	50
pp	*pole pairs*		2
Rs	*stator resistance*	[ohm]	6.41
Rr	*rotor resistance*	[ohm]	5.55
*Xsig_s_*	*stator leakage reactance at "Fno" Hz*	[ohm]	4.14
*Xsig_r_*	*rotor leakage reactance at "Fno" Hz*	[ohm]	7.03
*Xh_*	*mutual reactance at "Fno" Hz*	[ohm]	91.4
Jmot	*motor inertia*	[kg*m2]	0.0070
scorrNom	*nominal slip*		0.0800
fv	*viscous friction coefficient*	[Nms]	1.4E-3

Table App4. 1 - motor data

Calculated quantities:

$i_{Base} = 4.2\sqrt{2} = 5.93\ A$

$v_{Base} = 220\sqrt{2} = 311\ V$

$Xs\_ = Xsig\_s\_ + Xh\_ = 95.54\ ohm$ - *total stator reactance [ohm] at "Fno" Hz*

$xs = X_s \cdot \frac{i_{Base}}{v_{Base}} = 1.82\ p.u.$

$Xr\_ = Xsig\_r\_ + Xh\_ = 98.43\ ohm$ - *total rotor reactance [ohm] at "Fno" Hz*

$xr = X_r \cdot \frac{i_{Base}}{v_{Base}} = 1.879\ p.u.$

$xh = X_h \cdot \frac{i_{Base}}{v_{Base}} = 1.744\ p.u.$

$xsig\_r = xr - xh = 0.134\ p.u.$

$xsig\_s = xs - xh = 0.079\ p.u.$

$xsig\_r + xsig\_s = 0.123$; *(the total leakage reactance in p.u. , which is estimated)*

2. `ASJAPAN` motor

Vno	*ratedl voltage*	[V]	127
Ino	*6.4 rated current*	[A]	6.08
Fno	*rated frequency*	[Hz]	60
pp	*pole pairs*		2
*Rs_*	*stator resistance*	[ohm]	0.662
*Rr_*	*rotor resistance*	[ohm]	0.645
*Xsig_s_*	*stator leakage reactance at "Fno" Hz*	[ohm]	1.5072
*Xsig_r_*	*rotor leakage reactance at "Fno" Hz*	[ohm]	1.5072
*Xh_*	*mutual reactance at "Fno" Hz*	[ohm]	30.8976
Jmot	*motor inertia*	[kg*m2]	0.0617/6
scorrNom	*nominal slip*		0.0444
fv	*viscous friction coefficient*	[Nms]	1.4E-3

Table App4. 2 - motor data

3. `LAF2AWAX` motor

Vno	*rated voltage [rms]*	*[V]*	*117.00*
Ino	*rated current*	*[A]*	*3.30*
Fno	*rated frequency*	*[Hz]*	*100*
pp	*pole pairs*		*2*
Rs	*stator resistance*	*[ohm]*	*2.64*
Rr	*rotor resistance*	*[ohm]*	*2.77*
Lss	*stator leakage inductance*	*[H]*	*0.00605*
Lsr	*rotor leakage inductance*	*[H]*	*0.00532*
Lm	*magnetizing inductance*	*[H]*	*0.05952*
Jmot	*rotor inertia*	*[kg*m^2]*	*0.00038*

Table App4. 3 - motor data

$i_{Base} = 3.3\sqrt{2} = 4.67\ A$

$v_{Base} = 117\sqrt{2} = 165.46\ V$

$Ls = Lss + Lm = 0.06557\ H$ - *total stator inductance*

$Xs = L_S \cdot 2 \cdot \pi \cdot f = 41.199\ ohm$ - *total stator reactance at "Fno" Hz*

$xs = X_s \cdot \frac{i_{Base}}{v_{Base}} = 1.163\ p.u.$

$Lr = Lsr + Lm = 0.06484\ H$ - *total rotor inductance*

$X_r = L_r \cdot 2 \cdot \pi \cdot f = 40.74\ ohm$ - *total rotor reactance at "Fno" Hz*

$xr = X_r \cdot \frac{i_{Base}}{v_{Base}} = 1.15\ p.u.$

$Xh = Lm\ 2piw = 37.398\ ohm$

$xh = X_h \cdot \frac{i_{Base}}{v_{Base}} = 1.056\ p.u.$

$xsig\_r = xr - xh = 0.094\ p.u.$

$xsig\_s = xs - xh = 0.107\ p.u.$

$xsig\_r + xsig\_s = 0.201$ - the total leakage reactance in p.u.

$rs = R_S \cdot \frac{i_{Base}}{v_{Base}} = 0.0745\ p.u.$

$rr = R_R \cdot \frac{i_{Base}}{v_{Base}} = 0.0781\ p.u.$

4. `LAF1AWAX` motor

Vno	*rated voltage [rms]*	*[V]*	*230*
Ino	*rated current [rms]*	*[A]*	*1*
Fno	*rated frequency*	*[Hz]*	*75*
pp	*pole pairs*		*2*
Rs	*stator resistance*	*[ohm]*	*18.70*
Rr	*rotor resistance*	*[ohm]*	*19.70*
Lss	*stator leakage inductance*	*[H]*	*0.04520*
Lsr	*rotor leakage inductance*	*[H]*	*0.03860*
Lm	*magnetizing inductance*	*[H]*	*0.45100*
Jmot	*rotor inertia*	*[kg*m^2]*	*0.01250*

Table App4. 4 - motor data

$Ls = Lss + Lm = 0.4962\ H$ - total stator inductance

$Xs = L_S \cdot 2 \cdot \pi \cdot f = 233.83\ ohm$ - total stator reactance at "Fno" Hz

$xs = 1.01\ p.u.$

$Lr = Lsr + Lm = 0.4896\ H$ - total rotor inductance

$Xh = L_m \cdot 2 \cdot \pi \cdot f = 212.53\ ohm$

$xh = = 0.921\ p.u.$

$xsig\_r = xr - xh = 0.079\ p.u.$

$xsig\_s = xs - xh = 0.089\ p.u.$

$xsig\_r + xsig\_s = 0.168$ - the total leakage reactance in p.u.

$rs = 0.0813\ p.u.$

$rr = 0.0857\ p.u.$

APPENDIX 5 - *Test rig pictures*

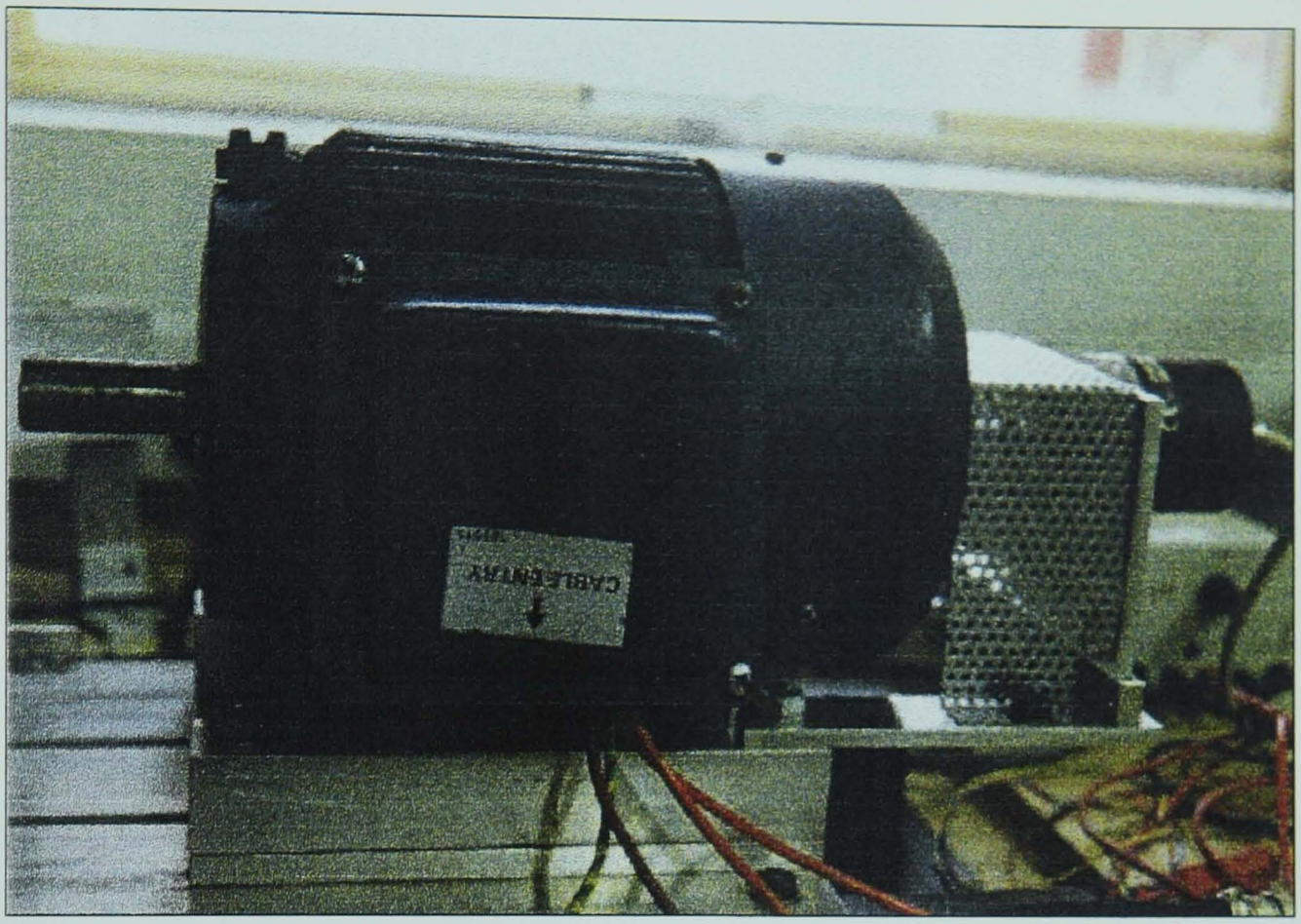

Figure app5.1- the 1.5 kW induction machine

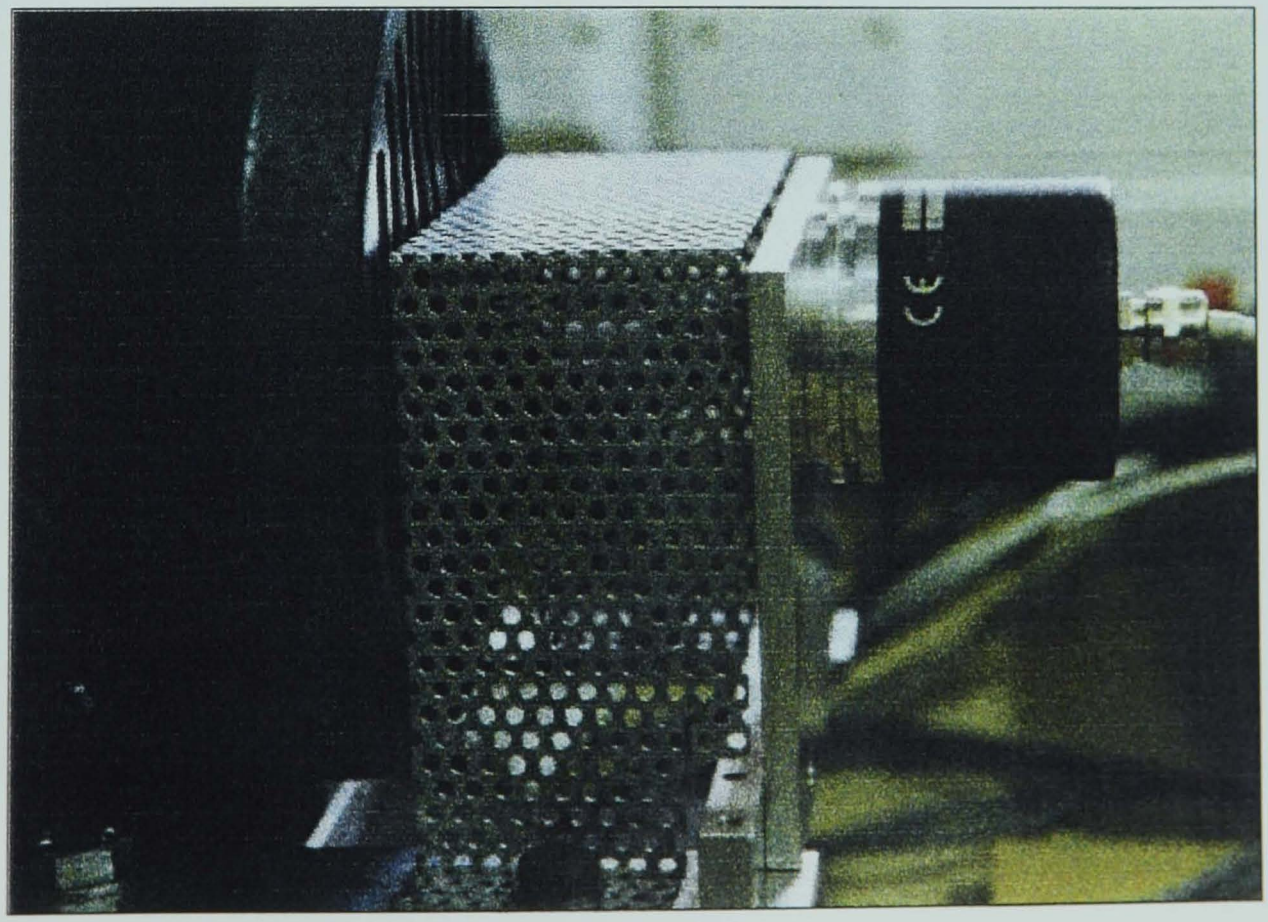

Figure app5.2- the encoder attached to the motor shaft

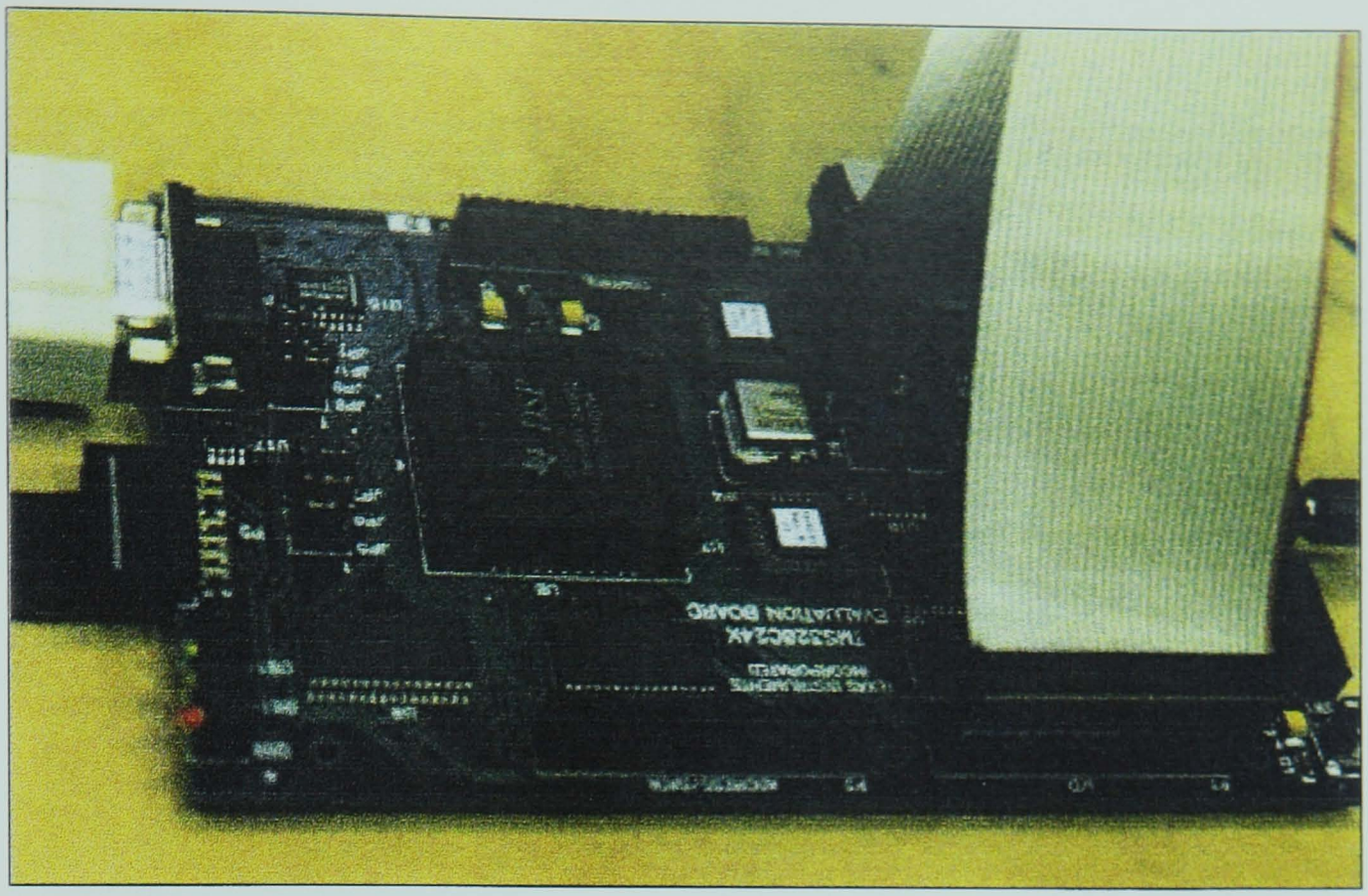

Figure app5.3- the Texas Instruments Evaluation Module (TMS320C240)

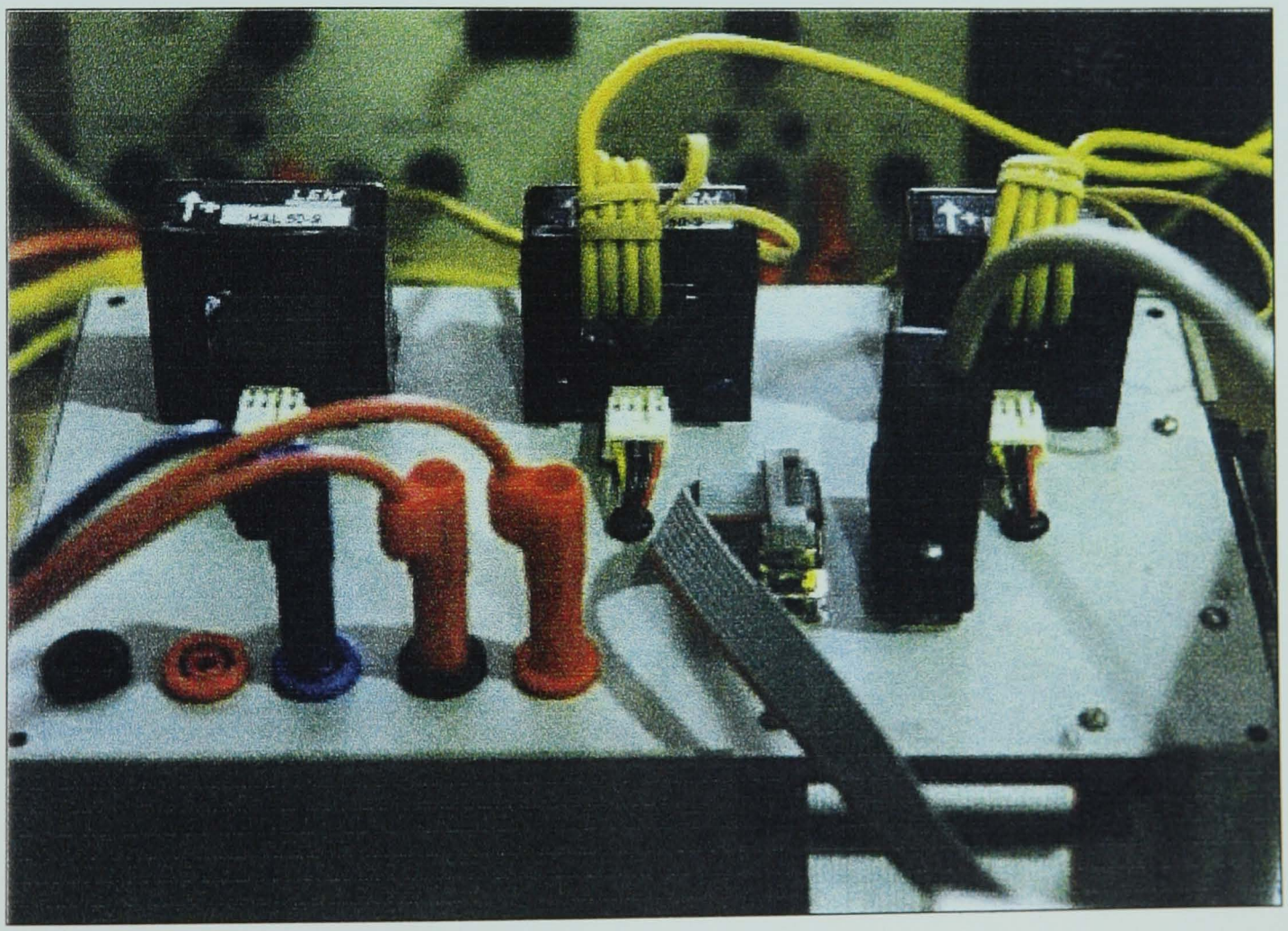

Figure app5.4- the interface between the encoder & Hall sensors and the DSP board

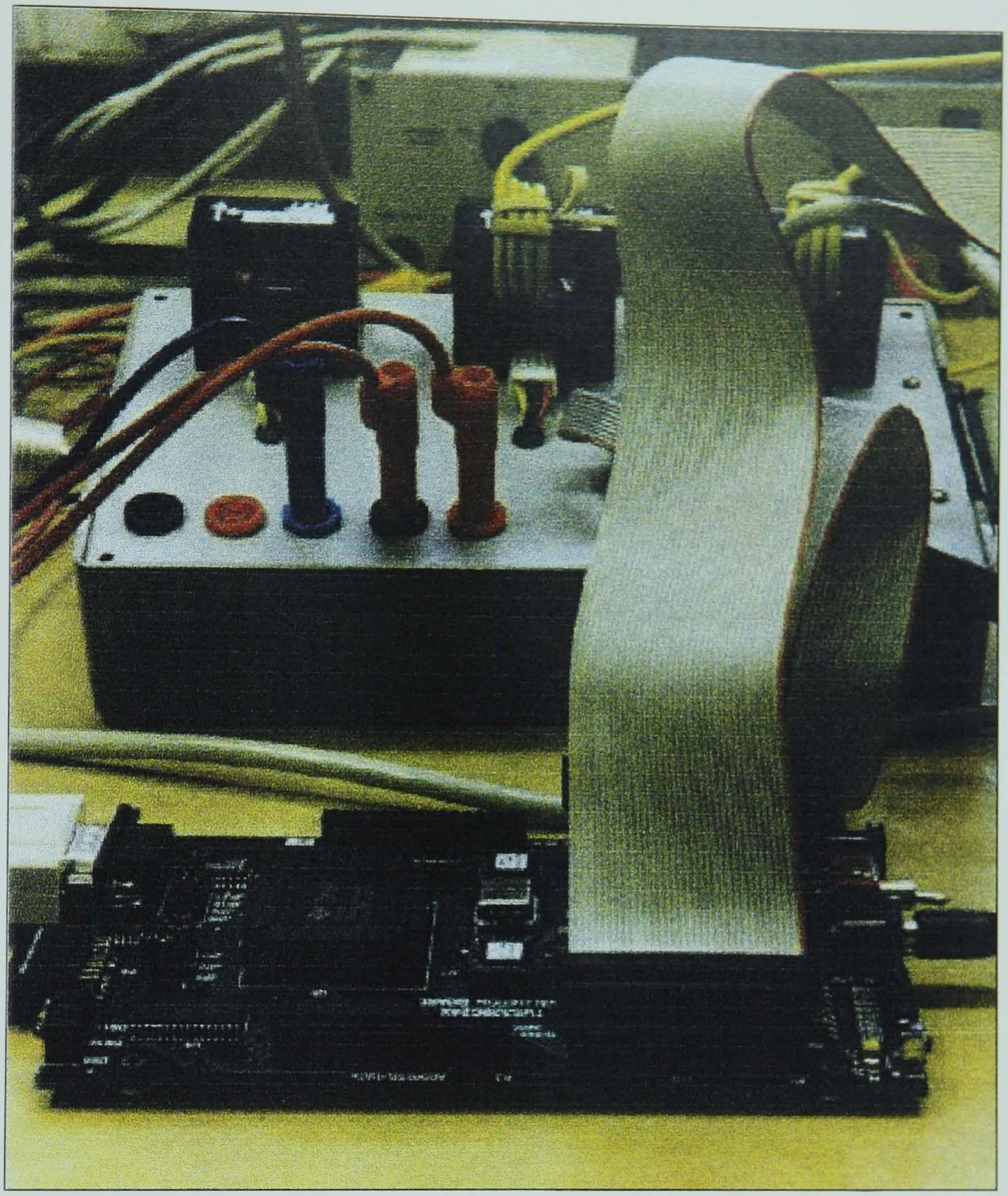

Figure app5.5- the interface connected to the DSP board

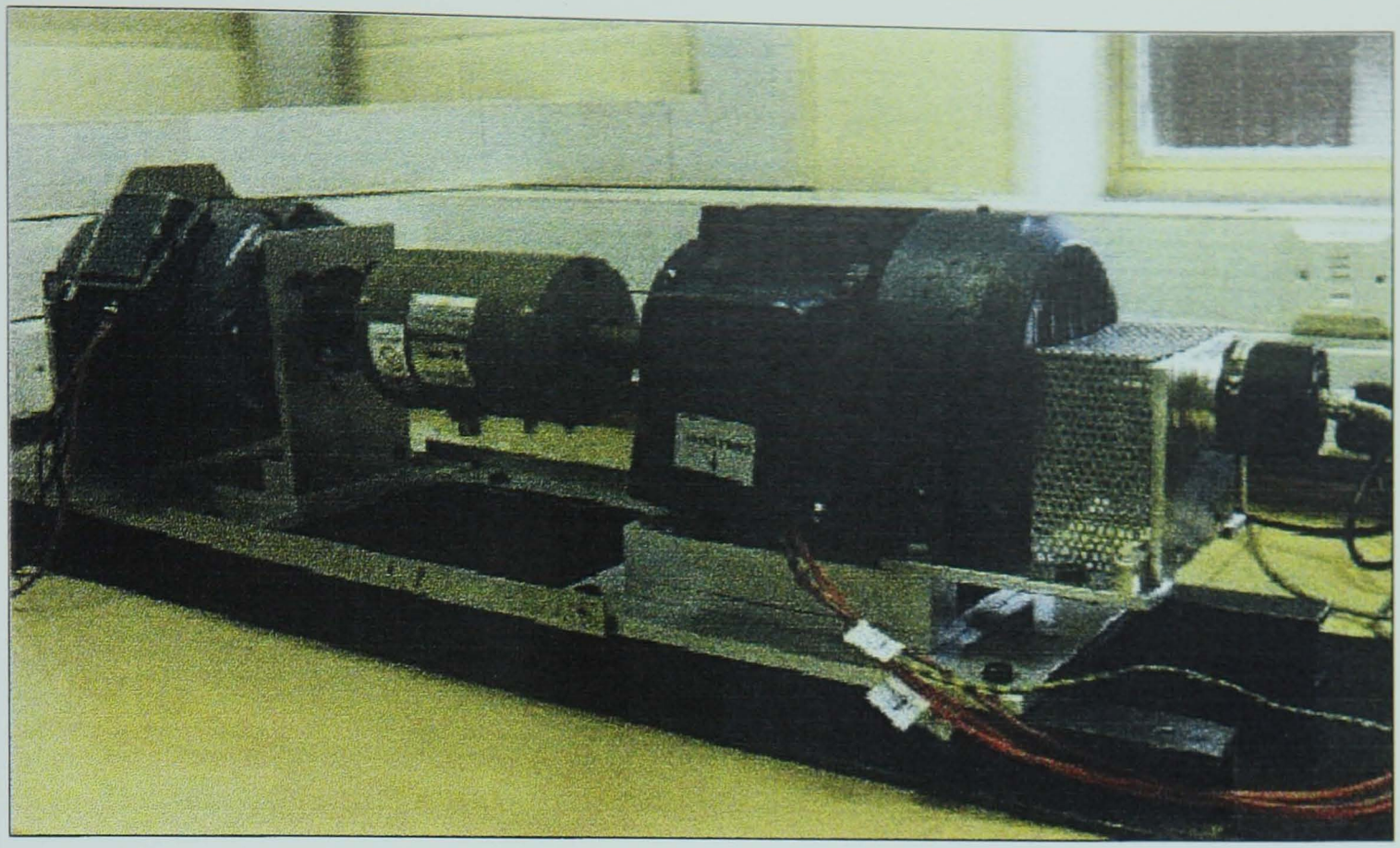

Figure app5.6- the test rig: d.c. machine, torque sensor and induction motor

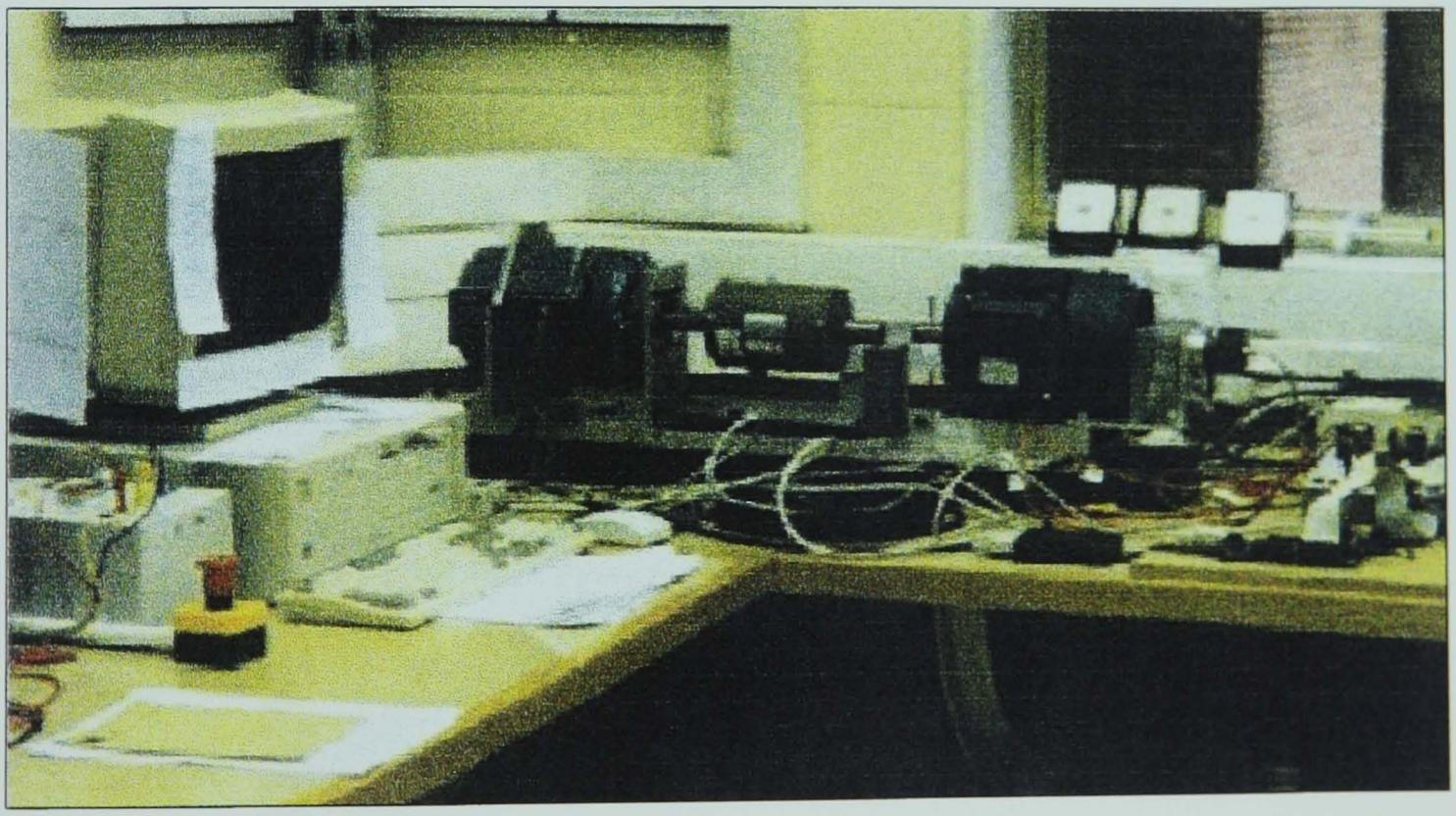

Figure app5.7 - the test-rig and the host PC

APPENDIX 6 - Assembler implemented control scheme (file: vec_new.asm)

```
;=================================================================
;------------file name vec_new.asm----------------------------
;---------based on vec_2.asm ------------------------------
;------------Anca Novinschi-----------------------------------
;-----------------------------------------------------------------
;=================================================================

            .include      "f240regs.h"
            .include      "tabsin.asm"
            .include      "tabteta.asm"
            .include      "tabvel4.asm"    ;for Tc=400us
            .include      "tabwti.asm"
            .copy        "variab.asm"

;=========================================================
ST0_save    .set   060h
ST1_save    .set   061h
ACCH        .set   062h
ACCL        .set   063h
ISRB        .set   7096h
;=========================================================

;----------------INITIALIZATIONS==============================

const1          .set   00DEh        ;sqrt(3)/2     in 8.8
const2          .set   0080h        ; 1/2 in 8.8
const3          .set   0094h        ; 1/sqrt(3) in 8.8
max_f           .set   0B33h        ; 0.7 pu in 4.12
min_f           .set   0F4CDh       ; -0.7 pu , in 4.12
max_ix          .set   0B33h        ; 0.7 pu in 4.12
min_ix          .set   0F4CDh       ; -0.7 pu , in 4.12
max_iy          .set   0B33h        ; 0.7 pu in 4.12
min_iy          .set   0F4CDh       ; -0.7 pu , in 4.12
min_spd         .set   0F4CDh
max_spd         .set   0B33h

;----------------------SAMPLING TIME-----------------

T1_per          .set   4000  ;T1 period, determining the PWM freq
T2_per          .set   4000  ;T2 period ,determining the sampling freq of the speed loop
T3_per          .set   60000  ;T3 period,for QEP circuit
tc              .set   8000  ; 8000 clock cycles for a sampling time of 400 us
to_min          .set   800
tctr            .set   14
wtr             .set   155

;----------transforming the currents after got them from ADC module------
;-----------------------------------------------------------------------
k1_transf_i     .set   0A08h  ;(2.5+0.0105)*1023=2568d=A08h
k2_transf_i     .set   0A03h
v_ref  .set    50Fh   ; 5.06 V in 8.8, supplied to the DSP board
kf     .set    350    ; 350d=0.085 in 4.12

                .data
first_chan      .word  CMPR2   ; for sectors: 3,1,5,4,6,2
                .word  CMPR1
                .word  CMPR1
                .word  CMPR3
                .word  CMPR2
                .word  CMPR3

second_chan     .word  CMPR1    ; for sectors: 3,1,5,4,6,2
                .word  CMPR3
                .word  CMPR2
                .word  CMPR2
                .word  CMPR3
                .word  CMPR1
;=========================================================
; Vector address declarations
;=========================================================
                .sect  ".vectors"

RSVECT          B   START     ; Reset Vector
```

```
INT1      B   INT1_ISR   ; Interrupt Level 1
INT2              B   ISR_A_T1C   ; Interrupt Level 2
INT3              B   ISR_B_T2U   ; Interrupt Level 3
INT4              B   PHANTOM   ; Interrupt Level 4
INT5              B   PHANTOM   ; Interrupt Level 5
INT6              B   PHANTOM   ; Interrupt Level 6
RESERVED          B   PHANTOM   ; Reserved
SW_INT8           B   PHANTOM   ; User S/W Interrupt
SW_INT9           B   PHANTOM   ; User S/W Interrupt
SW_INT10          B   PHANTOM   ; User S/W Interrupt
SW_INT11          B   PHANTOM   ; User S/W Interrupt
SW_INT12          B   PHANTOM   ; User S/W Interrupt
SW_INT13          B   PHANTOM   ; User S/W Interrupt
SW_INT14          B   PHANTOM   ; User S/W Interrupt
SW_INT15          B   PHANTOM   ; User S/W Interrupt
SW_INT16          B   PHANTOM   ; User S/W Interrupt
TRAP              B   PHANTOM   ; Trap vector
NMINT             B   PHANTOM   ; Non-maskable Interrupt
EMU_TRAP          B   PHANTOM   ; Emulator Trap
SW_INT20          B   PHANTOM   ; User S/W Interrupt
SW_INT21          B   PHANTOM   ; User S/W Interrupt
SW_INT22          B   PHANTOM   ; User S/W Interrupt
SW_INT23          B   PHANTOM   ; User S/W Interrupt
;=========================================================
; END of --Vector address declarations
;=========================================================


;=========================================================
; M A I N   C O D E  - starts here
;=========================================================

          .text
 START:       ; it comes here after an reset
 LDP  #00h
 SETC  INTM          ; disable interrupts, look at page 3-16, vol. I
 SPLK  #0h,IMR     ; mask all ints, page 6-18, vol.I
 SPLK  #0FFh,IFR     ; clear all ints flags, page 6-16,vol.I
 CLRC  SXM          ;clear sign extension mode , page 3-17 , vol. I
 CLRC  OVM          ; reset overflow mode , page 3-17 , vol. I
 CLRC  CNF          ;config block B0 of DARAM to data mem, page 3-16, vol. I

;----------watchdog----------vol.II , page 6-1-------
 LDP  #0E0H              ;memory page 224 (7000h-707F h)
 SPLK  #06Fh,WDCR
 SPLK  #55h,WDKEY
 SPLK  #0AAh,WDKEY

;----------CPU clock output mode------------
 SPLK  #0100000011000000b,SYSCR          ;SYSCR
 SPLK  #0000000000100000b,SYSSR          ;only bit 5 is set

;----------PLL clock module-------------------

 SPLK  #00BBh,  CKCR1       ; changed for SCI
 SPLK  #00000001b,CKCR0
 SPLK  #11000001b,CKCR0
 LDP  #0
 SPLK  #1000b,WSGR
 LDP  #0E1H                  ; page 225, addresses: 7080h-70FF
 SPLK  #0F800h,OCRA
 SPLK  #00F0h,OCRB
 SPLK  #0000h,PADATDIR
 SPLK  #0000h,PBDATDIR
 SPLK  #0000h,PCDATDIR

;----------------peripherals-----------------------------------
;---------------------------------------------------------------
 LDP  #0E0h                  ;back to page 224
 SPLK  #0000000000000100b,ADCTRL2
 SPLK  #0101100110101111b,ADCTRL1   ; means right  sensor and middle one

;===========EVENT MANAGER============================
;=====================================================

 LDP  #232                   ;addresses : 7400h-747Fh where EV is
 SPLK  #0,T1CON
 SPLK  #0,T2CON
 SPLK  #0,T3CON
 SPLK  #0,DBTCON
```

```
 SPLK #0,CAPCON
 SPLK #0,COMCON
 SPLK #0,T1CNT
 SPLK #0,T2CNT
 SPLK #0,T3CNT
 SPLK #T1_per,T1PR
 SPLK #T2_per,T2PR
 SPLK #T3_per,T3PR
;----------------ACTR register-------page 2-48, vol. II----------------------------

 SPLK #0000011001100110b,ACTR
 SPLK #0000011011111000b, DBTCON
 SPLK #0h,EVIMRA
 SPLK #0000001100000111b,COMCON
 SPLK #1000001100000111b,COMCON
 SPLK #0000000001010101b,GPTCON
 SPLK #98B2h, T3CON   ; T3CON[5-4]=11 as QEP circuit is the clock source
 SPLK  #1010100010000010b,T2CON
 SPLK  #1010100000000010b,T1CON
 SPLK #1110001011110000b,CAPCON

 ;flag registers
 SPLK #0FFFh,EVIFRA
 SPLK #0FFh,EVIFRB
 SPLK #0Fh,EVIFRC

 ;mask register
 SPLK #100h,EVIMRA
 SPLK #04h,EVIMRB
 SPLK #00h,EVIMRC

;==========END OF SETTINGS FOR EVENT MANAGER=========
;=====================================================


;*****************INITIALIZATIONS FOR SCI*************
;*****************************************************
SCI_INIT:
 LDP  #0E0h
 SPLK #0037h,SCICCR   ; odd parity
 SPLK #0013h,SCICTL1
 SPLK #0002h,SCICTL2
; baud rate
 SPLK #0000,SCIHBAUD
 SPLK #0082h,SCILBAUD       ; to have a baud rate of 9600
 SPLK #0022h,SCIPC2
 SPLK #0033h,SCICTL1
 LAR  AR0,#SCITXBUF
 LAR  AR1,#SCIRXBUF
 LAR  AR2, #B0_SADDR
 LAR  AR3, #end_prog
 LAR  AR4, #ACTR

;*****************END OF SCI INIT*******************
;*****************************************************

 LDP   #6
 SPLK #0,va
 SPLK #0,vb
 SPLK #0,vc
 SPLK #0,ia
 SPLK #0,ib
 SPLK #0,ic
 SPLK #0,ix
 SPLK #0,ix_ref
 SPLK #0,iy
 SPLK #0,iy_ref
 SPLK #03F9h ,vx_ref
 SPLK #0,vy_ref
 SPLK #0,vd_ref
 SPLK #0,vq_ref
 SPLK #0,buf_mul
 SPLK #0,tetar
 SPLK #0,sin_tetar
 SPLK #0,cos_tetar
 SPLK #0,wsync
 SPLK #0,sync_neg     ; flag to signal the negative wsync
 SPLK #0, tetar_neg  ; flag to signal the negative tetar
 SPLK #0,y_act
 SPLK #0,y_out
```

```
SPLK #0,y_dif
SPLK #0,err
SPLK #0,yf_prev
SPLK #0,yix_prev
SPLK #0,yiy_prev
SPLK #0,yspd_prev
SPLK #0,y_max
SPLK #0,y_min
SPLK #0,imag
SPLK #0,slip8
SPLK #0,slip12
SPLK #0,sva
SPLK #0,svb
SPLK #0,svc
SPLK #0,table_pointer
SPLK #0,temp_stor
SPLK #0,k1vq
SPLK #0,k2vd
SPLK #0,k3vq
SPLK #0,teta
SPLK #0,teta_old
SPLK #0,d_teta
SPLK #0,wrot
SPLK #0,i1
SPLK #0,i2
SPLK #0,i1i2
; SPLK #05h,cont_20   ;for estimating speed
 SPLK #0Ah,cont_20
 SPLK #0,ti
 SPLK #0,tj
 SPLK #0,to
 SPLK #0,sv_bits
 SPLK #0,sett
 SPLK #0,sett_real
 SPLK #0,start_sampl
 SPLK #0,id
 SPLK #0,iq
 SPLK #0, tetar_found
 SPLK #0, counter_tc
 SPLK #0, flag_wrot
 SPLK #7698,k1     ; for 0.9
 SPLK #13333,k2
 SPLK #15396,k3

;----------REGULATORS COEFFICIENTS----------

 SPLK #3669, kpif
 SPLK #30, kif
 SPLK #34, kcorf

 SPLK #3522, kpiix
 SPLK #434, kiix
 SPLK #505, kcorix

 SPLK #3522, kpiiy
 SPLK #434, kiiy
 SPLK #505, kcoriy

 SPLK #18743, kpispd
 SPLK #130, kispd
 SPLK #50, kcorspd
;-------------------------------------------------------

 SPLK #0,test_w
 SPLK #0, zero
 SPLK #0,save1_count
 SPLK #0,save2_count
 SPLK #0,save3_count
 SPLK #0,save5_count
 SPLK #0,tr1_count
 SPLK #0, acc_h
 SPLK #0, acc_l
 SPLK #0FFFFh, nr_ffff
 SPLK #0, readng
 SPLK #0, data_end
 SPLK #0FFCh .const_4mii
 SPLK #2Dh ,const_45
 SPLK #0h, tr3_count
 SPLK #0h,wrot_f
 SPLK #0h, comp_0
```

```
 SPLK #0h, comp_1
 SPLK #0h, comp_2
 SPLK #0h, chan_1
 SPLK #0h, chan_2
 SPLK #0h, spd_ref
 SPLK #0h, begin_flag
 SPLK #0400h, imag_ref  ; 0400h= 0.25 pu in 4.12

;-------------VARIABLES-----------------------------
;-----------INIT  TABLE FIRST ENTRIES----------------
;=========================================================
 SPLK #SIN_1,sin_first
 SPLK #COS_1,cos_first
 SPLK #TETA_1,theta_first
 SPLK #TVEL_1,vel_first
 SPLK #TWTI_1,twti_first
;=========================================================
 SPLK #end_prog, prog_add
 SPLK #XDATA_SADDR, ext_mem

                .text

;============ MAIN=====================================
 LAR  AR6, #B0_EADDR ;  load in AR6 end address of B0 , 02FFh
 MAR  *,AR6
 SPLK #0,*
 LDP  #6
 LACL prog_add
 ADD  #0FBh
 SACL data_end
 TBLW zero
 LAR  AR6, data_end
 LAR  AR7, #XDATA_SADDR

 LDP  #0
 SPLK #0FFh,IFR          ; clear flags
 SPLK #07h,IMR
 EINT
 LDP#232
 SPLK #1010100001000010b,T1CON      ;starts the timers

BEGIN_SMPL:
 LDP  #6
wait_loop:
 LACC start_sampl
 BCND wait_loop, EQ
 SPLK #0, start_sampl

;==============GETTING THE SPEED FROM ENCODER=================

 LDP  #232
 LACL T3CNT
 LDP  #6
 SACL teta
 LACL cont_20
 SUB  #1
 BCND time_for_acq, EQ           ; if ACC =0
 SACL cont_20
 B    no_acq
time_for_acq:
 SPLK #1, flag_wrot ; to signal that wrot was estimated
 LACL teta
 SUB  teta_old
 SACL buf_mul
 LACC buf_mul,16
 BCND rotat_pos,GEQ
 NEG
rotat_pos:
 SACH buf_mul
 LACC buf_mul,14    ;divide by 4
 SACH d_teta
 SACH wrot
 SPLK #10,cont_20
 LACL teta
 SACL teta_old
no_acq:

;=========================================================
;----GETTING CURRENTS FROM ADC module----------
;-------------------------------------------------------
```

```
adc_currents:
 LDP  #0E0h
 LACC ADCFIFO1,10   ; get data from ADC , page 3-10,vol.II
 LDP  #6
 SACH i1  ;if ADCFIFO1 is the data register of ADC1, then i1 is the current
 LDP  #0E0h
 LACC ADCFIFO2,10
 LDP  #6
 SACH i2   ; this current comes from the Hall sensor in the middle

;=======restart ADC ======================
 LDP  #0E0h
SPLK #0101100110101111b,ADCTRL1 ; means right  sensor and middle one
;================ADC restarted================

transf_i1:
 LDP  #6
 LT   i1
 MPY  #v_ref     ; is 5.1 V , the voltage supplied to DSP board
 SPM  1
 PAC
 SACH i1,7
 LACC #k1_transf_i
 SUB  i1
 SACL i1
 LT   i1
 MPY  #194h
 PAC
 SACH i1,7
;----------------i1 is now evaluated---is in 4.12 format-------
;-------------------------------------------------------------

;--------now scaling i2-----------
transf_i2:
 LDP  #6
 LT   i2
 MPY  #v_ref     ; is 5.1 V , the voltage supplied to DSP board
 PAC
 SACH i2,7
 LACC #k2_transf_i
 SUB  i2
 SACL i2
 LT   i2
 MPY  #194h
 PAC
 SACH i2,7

 ;===now i1 and i2 are available, they are in pu and in 4.12 format=========
 ;----END OF GETTING CURRENTS FROM ADC module----------
 ;=========================================================

 ;=========================================================
 ; ----conversion ia, ib, ic ==>  id, iq****id=i1**iq=(i1+2*i2)/sqrt(3)
 ;=========================================================

 abc_dq:
  LACC i2, 1      ;load low ACC with value of i2, left  shifted by 1 position
  ADD  i1         ; in ACC <== i1+2*i2
  SACL i1i2
  LT   i1i2       ;load TREG in prep for a multiplication
  MPY  #const3       ;  const3=1/sqrt(3) const3 in 8.8
  SPM  1          ; PREG output to be shifted 1 places
  PAC
  SACH iq,7         ; iq in 4.12 , logbook p.153
  LACL i1           ;  i1 in 4.12
  SACL id           ;store low ACC in id , id in 4.12
 ; ----at this point id, iq calculated, in 4.12 format-----------

 ;=========================================================
 ; ----use ix, iy from the previous Tc----------------
 ; ----from:     Tr*(dimag/dt)+imag=ix  ==>
 ;-----------------------------------------------------
 ;---------------imag=imag_old+Tc/Tr*(ix-imag_old)------
 ;-----------------------------------------------------
 ;=========================================================
calc_imag:
 LACL ix           ; ix in 4.12
 SUB  imag           ;  imag in 4.12
 SACL buf_mul
```

```
LT   buf_mul        ; buf_mul , 4.12
MPY  #tctr          ; tctr * buf_mul= Tc/Tr*(ix-imag)*** tctr in 4.12
SPM  2             ; look in the logbook, p87 , p153
PAC
SACH buf_mul        ;working in 4.12
LACL buf_mul        ; 4.12
ADD  imag          ; 4.12
SACL imag           ; imag=imag_old+tctr*(ix-imag)
SACL buf_mul

;=============================================================
;------------SLIP CALCULATION----------------------------
;     the formula used :  wmr=wr+iy/(Tr*imag)       /wsync
;-------------------------------------------------------
;      ==>  wmr/wsync=wr/wsync+iy/(Tr*imag*wsync)=slip=s
;-------------------------------------------------------
;     1/(Tr*wsync)=a constant=wTr (for ex)
;=============================================================

slip_calc:
 SETC  SXM
 LACC  buf_mul    ; in buf_mul is imag
 BCND  imag_pos , GEQ
 NEG
 SACL  buf_mul

imag_pos:
 CLRC SXM
 LACC buf_mul,12       ; in buf_mul is imag
 SACH  table_pointer
 LACC  twti_first
 ADD   table_pointer
 TBLR  temp_stor
 LT    temp_stor      ;got a value from the table and load TREG with that, is in 4.12
 MPY   iy            ;multiply iy*wTr/imag, result is slip in PREG, iy in 4.12
 SPM   0
 PAC
 SACH  slip8         ;direct in 8.8 , corect
 LACL  slip8
 ADD   wrot          ;wrot is the measured speed, in pu, 8.8
 SACL  wsync         ;the sync speed
 SETC  SXM
 LACC  wsync         ; check if sync speed is neg
 BCND  wsync_pos, GEQ
 SPLK  #1,sync_neg
 NEG
 SACL  wsync

wsync_pos:
 CLRC  SXM
 LACL  wsync
 SACL  table_pointer
 LACL  vel_first           ;  the address of the first entry (TVEL_1) is loaded in ACC
 ADD   table_pointer
 TBLR  temp_stor       ;in temp_stor is now Dtetar , in 12.4 format
 LACL  sync_neg
 SUB   #1
 BCND  nochange, LT
 LACL  tetar
 SUB   temp_stor
 SACL  tetar  ;if wsync was negative , then tetar(new)=tetar-Dtetar
 B     tetar_off

nochange:
 LACL  tetar         ;the old tetar
 ADD   temp_stor           ; in ACC now tetar+Dtetar
 SACL  tetar

tetar_off:
 SETC  SXM
 LACC  tetar       ; check if tetar is neg
 BCND  tetar_pos, GEQ
 SPLK  #1,tetar_neg    ;flag set if tetar was negative
 NEG
 SACL  tetar
 SUB   #5760
 BCND  tetar_ok,LT
 SACL  tetar
 B     tetar_ok
tetar_pos:
```

```
 SUB  #5760
 BCND  tetar_ok,LT    ; ----if the result is positive
 SACL  tetar
tetar_ok:
 CLRC  SXM
; ----if result is negative then nothing , tetar was < 360 , OK

;=======================================================
;--------GENERATING sin(tetar) and cos(tetar)----------
;=======================================================

 LACL  tetar
 SUB  #5h
 BCND  tetar_six,GT
 SPLK  #0, sin_tetar
 SPLK  #0100h, cos_tetar
 B    after_sin

tetar_six:
 LT   tetar
 MPY  #0008h   ;tetar/(360*16/1024)  = tetar*(8/45)
 SPM  0
 PAC
 RPT  #15
 SUBC  const_45
 SUB  #1
 SACL  table_pointer

ok_pointer:
 ADD  theta_first
 TBLR  tetar_found
 CLRC  SXM

goto_sin:
          LACL   sin_first
          ADD    table_pointer
          TBLR   sin_tetar
          LACL   cos_first
          ADD    table_pointer
          TBLR   cos_tetar

after_sin:
 LACL  tetar_neg   ; the flag for negative tetar
 SUB  #1
 BCND  sincos_ok, LT
 LACL  sin_tetar    ; if tetar was negative then sin_tetar is neg
 NEG
 SACL  sin_tetar
 LACL  tetar
 NEG
 SACL  tetar

;=======for when speed reference is zero============
sincos_ok:
 LACL  spd_ref         ;DEBUG
 BCND  dq_xy,NEQ
 SPLK  #0h, sin_tetar
 SPLK  #0100h, cos_tetar
 SPLK  #0, iq
dq_xy:
;-------------------------------------------------------
;----TRASFORMATION----- D, Q ===> X, Y----------
;-------------------------------------------------------
;   ix = id*cos(tetar) + iq*sin(tetar)
;   iy = -id*sin(tetar) + iq*cos(tetar)
;-------------------------------------------------------
  LT   id          ; id in4.12
  MPY  cos_tetar           ; id* cos_tetar, cos_tetar in 8.8
  SPM  1           ; that means PM=01
  PAC
  SACH  buf_mul, 7     ; 7+1 gives a total shift of 8 positions
  LT   iq          ; 4.12
  MPY  sin_tetar      ; 8.8
  PAC
  SACH  ix, 7         ; ix in 4.12

  LACL  ix
  ADD  buf_mul
  SACL  ix          ; ix = id*cos(tetar) + iq*sin(tetar) , ix in 4.12
;-------------------------------------------------------
```

```
 LT   id          ; 4.12
 MPY  sin_tetar          ; id* sin_tetar , sin_tetar in 8.8
 PAC
 SACH buf_mul, 7
 LT   iq          ; 4.12
 MPY  cos_tetar          ; 8.8
 PAC
 SACH iy, 7       ; iy in 4.12
 LACL iy    ;
 SUB  buf_mul
 SACL iy          ; iy = -id*sin(tetar) + iq*cos(tetar)  in 4.12
;-------------------------------------------------------------------------
; NOW ix, iy ARE AVAILABLE ; THEY ARE IN 4.12 FORMAT

;-------------------------------------------------------------------
;----------------FLUX REGULATOR ( PI type)
;-------------------------------------------------------------------
fluxreg:
 LACL imag_ref        ;DEBUG
 BCND filtro ,EQ
 LACC imag_ref              ;4.12
 SUB  imag            ;4.12
 SACL err             ;4.12
 LACC #max_f            ; in 4.12
 SACL y_max
 LACC #min_f            ; in 4.12
 SACL y_min
 CALL PIREG_F      ; page 8-58 vol .I, RET is at 8-141
 LACL y_out
 SACL ix_ref           ; result in 4.12

;-------------------------------------------------------------------
;----------------ix CURRENT REGULATOR ( PI type)
;-------------------------------------------------------------------
ixreg:
 LACL ix_ref          ;4.12
 SUB  ix             ;4.12
 SACL err             ;4.12
 LACC #max_ix
 SACL y_max
 LACC #min_ix
 SACL y_min
 CALL PIREG_IX          ; page 8-58 vol .I, RET is at 8-141
 LACL y_out
 SACL vx_ref           ; result in 4.12

;-------------------------------------------------------------------
;----------------SPEED REGULATOR ( PI type)
;-------------------------------------------------------------------
filtro:
 LACC spd_ref
 BCND speedreg, EQ
 SUB  wrot_f
 SACL buf_mul
 LT   buf_mul
 MPY  #kf
 SPM  2
 PAC
 SACH buf_mul
 LACL buf_mul
 ADD  wrot_f
 SACL wrot_f  ; reference speed in 8.8 , comes from the filter
speedreg:
 SUB  wrot            ; rotor speed , comes from the encoder
 SACL err,4       ;4.12
 LACC #max_spd
 SACL y_max
 LACC #min_spd
 SACL y_min
 CALL PIREG_SPD
 LACL y_out
 SACL iy_ref

;-------------------------------------------------------------------
;----------------iy CURRENT REGULATOR ( PI type)
;-------------------------------------------------------------------
iyreg:
 LACL iy_ref          ;4.12
 SUB  iy             ;4.12
 SACL err             ;4.12
```

```
 LACC #max_iy
 SACL y_max
 LACC #min_iy
 SACL y_min
 CALL PIREG_IY
 LACL y_out
 SACL vy_ref              ; vy_ref in 4.12

;-------------------------------------------------------
;----TRASFORMATION----- X, Y ===> D, Q ----------------
;-------------------------------------------------------
;   vd_ref = vx_ref * cos(tetar) - vy_ref * sin(tetar)
;   vq_ref = vx_ref * sin(tetar) + vy_ref * cos(tetar)
;-------------------------------------------------------

xy_dq:
 LT   vx_ref       ; vx_ref in 4.12
 MPY  cos_tetar            ; vx_ref * cos_tetar, cos_tetar in 8.8
 SPM  1
 PAC
 SACH buf_mul, 7     ; buf_mul in 4.12

 LT   vy_ref       ; vy_ref in 4.12
 MPY  sin_tetar            ; sin_tetar in 8.8
 PAC
 SACH vd_ref, 7            ; vd_ref in 4.12
 LACL buf_mul
 SUB  vd_ref
 SACL vd_ref       ; vd_ref = vx_ref * cos(tetar) - vy_ref * sin(tetar)
;-------------------------------------------------------------------------
 LT   vx_ref       ; vx_ref in 4.12
 MPY  sin_tetar            ; vx_ref* sin_tetar
 ;SPM  1
 PAC
 SACH buf_mul, 7
 LT   vy_ref       ; vy_ref in 4.12
 MPY  cos_tetar            ; cos_tetar in 8.8
 PAC
 SACH vq_ref, 7
 LACL buf_mul
 ADD  vq_ref
 SACL vq_ref       ; vq_ref = vx_ref * sin(tetar) + vy_ref * cos(tetar)
;-------------------------------------------------------------------------
;----END OF TRASFORMATION----- X, Y ===> D, Q --------
;-------------------------------------------------------------------------

;-------------------------------------------------------------------------
;----------- TRASFORMATION----- D, Q ===>A , B , C------
;-------------------------------------------------------------------------
;      Va=Vq_ref
;      Vb= - 1/2 * Vq_ref + sqrt(3)/2 * Vd_ref
;      Vc= - 1/2 * Vq_ref - sqrt(3)/2 * Vd_ref
;-------------------------------------------------------------------------

dq_abc:
 LACL vq_ref              ; vq_ref is in 4.12
 SACL va                 ; Va=Vq_ref
 LT   vq_ref             ; vq_ref is in 4.12
 MPY  #const2              ; const2=1/2 in 8.8
 SPM  1
 PAC
 SACH vb, 7               ; vb in 4.12
 LT   vd_ref             ; vd_ref is in 4.12
 MPY  #const1              ; const1=sqrt(3)/2 in 8.8
 ;SPM  1
 PAC
 SACH buf_mul, 7           ; buf_mul in 4.12
 LACL buf_mul
 SUB  vb
 SACL vb
 LT   vq_ref             ; vq_ref is in 4.12
 MPY  #const2
 PAC
 SACH vc, 7               ; vc in 4.12
 LACL buf_mul
 ADD  vc
 NEG
 SACL vc
;-------------------------------------------------------------------------
;----END OF TRASFORMATION----- D, Q ===> A, B, C -------
```

```
;----va , vb, vc in 4.12----------
;-------------------------------------------------------------------------

;-------------------------------------------------------------------------
;----SIGN OF Va, Vb, Vc----------
;-------------------------------------------------------------------------
            SETC   SXM
            LACC   va
            BCND   neg_a, LT
            SPLK   #1,sva
            B      sign_b
neg_a:
            SPLK   #0, sva
sign_b:
            LACC   vb
            BCND   neg_b, LT
            SPLK   #1, svb
            B      sign_c
neg_b:
            SPLK   #0, svb
sign_c:
            LACC   vc
            BCND   neg_c, LT
            SPLK   #1, svc
            B      sign_end
neg_c:
            SPLK   #0, svc
sign_end:
            CLRC   SXM
;-------------------------------------------------------------------------
;----SECTOR CALCULATION ----------
;-------------------------------------------------------------------------
; sector 1 is coded 3, 2->1 , 3->5, 4->4, 5->6, 6->2
; sett=sva+2*svb+4*svc        logbook , page 149-150
;-------------------------------------------------------------------------
 LACC  svc, 2              ; in ACC is now 4*svc
 ADD   sva
 SACL  sett            ; sett=4*svc +sva
 LACC  svb,1               ; in ACC is now 2*svb
 ADD   sett                ;
 SACL  sett            ; sett=2*svb+4*svc +sva
end_sett:
;-------------------------------------------------------------------------
;----------Ti, Tj , To EVALUATION----------Ti+Tj+To=Tc (sampling step)
;-------------------------------------------------------------------------
; vd_ref and vq_ref are in 4.12

 LT   vd_ref           ; 4.12
 MPY  k2               ; an integer
 SPM  0
 PAC
 SACH k2vd,4             ; k2vd is an integer
 LT   vq_ref           ; 4.12
 MPY  k1               ; an integer
 PAC
 SACH k1vq,4             ; k1vq is an integer
 LT   vq_ref           ; 8.8
 MPY  k3               ; an integer
 PAC
 SACH k3vq,4             ; k3vq is an integer

;---sector determination by looking at tetar---------
;-------------------------------------------------
sector_determ:
 MAR  *,AR5
 LAR  AR5,#1
 LACC tetar
 SUB  #960       ;60degrees in 12.4 (sector1)
 BCND end_setfind,LEQ
 MAR  *+
 LACC tetar
 SUB  #1920      ;120degrees in 12.4 (sector2)
 BCND end_setfind,LEQ
 MAR  *+
 LACC tetar
 SUB  #2880      ;180degrees in 12.4 (sector3)
 BCND end_setfind,LEQ
 MAR  *+
 LACC tetar
 SUB  #3840      ;240degrees in 12.4 (sector4)
```

```
 BCND end_setfind,LEQ
 MAR  *+
 LACC tetar
 SUB  #4800       ;300degrees in 12.4 (sector5)
 BCND end_setfind,LEQ
 MAR  *+
end_setfind:
 SAR  AR5, sett_real      ;the real numerotation 1,2,3,4,5,6
;=======end of determination of sector by tetar================

 LACL spd_ref   ;debug
 BCND sect_1,EQ  ;this goes to sector 1 if spd_ref is 0
 LACL sett
 SUB  #1
 BCND sect_2,EQ
 LACL sett
 SUB  #2h
 BCND sect_6,EQ
 LACL sett
 SUB  #3h
 BCND sect_1,EQ
 LACL sett
 SUB  #4h
 BCND sect_4,EQ
 LACL sett
 SUB  #5h
 BCND sect_3,EQ
 LACL sett
 SUB  #6h
 BCND sect_5,EQ
sect_1:                ; the real sector 1, coded: 3
          LACL  k2vd
          SUB   k1vq
          SACL  ti          ; ti is an integer
          LACL  k3vq
          SACL  tj
          B     sect_end
 sect_2:
          LACL  k1vq
          SUB   k2vd
          SACL  ti
          LACL  k1vq
          ADD   k2vd
          SACL  tj
          B     sect_end
sect_3:
          LACL  k3vq
          SACL  ti
          LACL  k1vq
          ADD   k2vd
          NEG
          SACL  tj
          B     sect_end
sect_4:
          LACL  #0
          SUB   k3vq
          SACL  ti
          LACL  k1vq
          SUB   k2vd
          SACL  tj
          B     sect_end
 sect_5:
          LACL  k1vq
          ADD   k2vd
          NEG
          SACL  ti
          LACL  k2vd
          SUB   k1vq
          SACL  tj
          B     sect_end
 sect_6:
          LACL  k1vq
          ADD   k2vd
          SACL  ti
          LACL  #0
          SUB   k3vq
          SACL  tj
          B     sect_end
 sect_end:
 ;-----------------------------------------------------------------
```

```
;----------END OF ----Ti, Tj -----EVALUATION----------
;-------------------------------------------------------------------------------

;-----------------------------------------------------------------
;---------- To---- EVALUATION----------Ti+Tj+To=Tc (sampling step)------
;---REEVALUATION of ti,tj,to if to is too small----------
;-----------------------------------------------------------------
 LACC #tc            ; tc=sampling step
 SUB  ti
 SUB  tj             ; now in ACC is to=Tc-ti-tj
 SUB  #to_min        ; to_min=10us
 BCND to_ok,GEQ
to_mic:
          LACL  ti
          ADD   tj
          SACL  buf_mul
          LACC  #tc
          SUB   #to_min
          SACL  to
          LT    to
          MPYU  ti          ; integer
          SPM   0
          PAC
          RPT   #15
          SUBC  buf_mul
          SACL  ti
          LACC  #tc
          SUB   #to_min
          SUB   ti
          SACL  tj
          LACC  #to_min
          SACL  to
          B     to_end
to_ok:
          ADD   #to_min
          SACL  to
to_end:
;--------end of re-evalation----------
;=================================================================
;--------conversion of ti,tj,to to CMPR1, CMPR2, CMPR3 ----------
;-----------------------------------------------------------------
;---example:----------compare register1= To/4
;----------compare register2= To/4+T1/2
;----------compare register2= To/4+T1/2+T2/2
;=================================================================

 CLRC SXM
 LACC to,14
 SACH comp_0   ; in comp_0=to/4
 LACC ti,15
 SACH comp_1   ; in comp_1=ti/2
 LACC tj,15
 SACH comp_2   ; in comp_2=tj/2
 LACL spd_ref
 BCND toggle_chan,NEQ
 SPLK #3, sett

toggle_chan:
 LACC #(first_chan-1)
 ADD  sett
 TBLR chan_1
 LAR  AR5, chan_1
 LACC comp_0
 SACL *            ; to/4 is saved in the CMPR register
 LACC #(second_chan-1)
 ADD  sett
 TBLR chan_2
 LAR  AR5, chan_2
 LACC comp_0
 ADD  comp_1
 SACL *            ; (to/4+ti/2) is saved in the CMPR register
 LACC #CMPR3
 SUB  chan_1
 ADD  #CMPR2
 SUB  chan_2
 ADD  #CMPR1
 SACL buf_mul      ; the address of the last channel to toggle
 LAR  AR5, buf_mul
```

```
 LACC  comp_0
 ADD   comp_1
 ADD   comp_2
 SACL  *          ; (to/4+ti/2+tj/2) saved to the CMPR register
;=========END OF WRITING TO CMPR1 AND CMPR2================
;===========================================================

 B     BEGIN_SMPL
            .copy "regs.asm"
            .text

;=============================================================================
; I S R_B_T2U
; Description:  ISR, used to handle the interrupt----------
; coming from timer T2, on underflow-----------------------
; Timer T2 counts from 0 to 2000 (4000) and down to 0 ,----------
; all this represents the sampling period.-----------------------
;=============================================================================

ISR_B_T2U:

 SST  #0,96       ; 96=60h, and status register ST0 is saved at 60h
 SST  #1,97       ; ST1 is saved at 61h
 LDP  #0
 SACH ACCH        ; saves High ACC at address ACCH
 SACL ACCL

 LDP  #232
; the Event manager registers page addresses: 7400h-747Fh
 LACC EVIFRB
 SACL EVIFRB
 LACC EVIFRB
 LDP  #6
 SPLK #1, start_sampl
 LDP  #0
 LACC ACCH,15
 SFL
 ADDS ACCL        ; ACC is restored
 LST  #1,61h      ; restore status register ST1
 LST  #0,60h
 EINT
 RET
; -------END OF---I S R_B_T2U----------
;=============================================================================


;=============================================================================
; I S R  -  PHANTOM
; Description:  Dummy ISR, used to trap spurious interrupts.
; Modifies:    Nothing
;=============================================================================
PHANTOM:

 SST  #0,96       ; 96=60h, and status register ST0 is saved at 60h
 SST  #1,97       ; ST1 is saved at 61h
 LDP  #0
 SACH ACCH        ; saves High ACC at address ACCH
 SACL ACCL

 LDP  #0
 LACC ACCH,15
 SFL
 ADDS ACCL        ; ACC is restored
 LST  #1,61h      ; restore status register ST1
 LST  #0,60h
 EINT
 RET
;=============================================================================
```